Near-Optimal Sublinear Time Algorithms for Ulam Distance

Alexandr Andoni* Huy L. Nguyen†

Abstract

We give near-tight bounds for estimating the edit distance between two non-repetitive strings (Ulam distance) with constant approximation, in sub-linear time. For two strings of length d and at edit distance R, our algorithm runs in time $\tilde{O}(d/R+\sqrt{d})$ and outputs a constant approximation to R. We also prove a matching lower bound (up to logarithmic terms). Both upper and lower bounds are improvements over previous results from, respectively, [Andoni-Indyk-Krauthgamer, SODA'09] and [Batu-Ergun-Kilian-Magen-Raskhodnikova-Rubinfeld-Sami, STOC'03].

1 Introduction

The edit distance (aka *Levenshtein distance*) between two strings A and B, denoted $\mathrm{ed}(A, B)$, is the minimum number of character insertions, deletions, and substitutions needed to transform one string into the other. This distance is of key importance in several fields such as computational biology and text processing, and consequently computational problems involving the edit distance were studied quite extensively.

The Ulam metric is a specialization of edit distance to non-repetitive strings, where a string is *non-repetitive* if every symbol appears at most once in it. There are several motivations for studying this variant. From a practical perspective, strings with limited or no repetitions appear in several important contexts, such as ranking of objects such as webpages (see, e.g., [AJKS02] and [Mar95]).

From a theoretical point of view, Ulam metric presents a concrete waypoint towards the elusive goal of designing algorithms for edit distance over general (or even binary) strings. Indeed, there are two reasons for this. First, Ulam metric appears to retain one of the core difficulties of the edit distance on general strings, namely the existence of "misalignments" between the two strings. In fact, there is no known lower bound that would strictly separate general edit distance from Ulam metric: all known lower bounds are nearly the same (quantitatively) for both metrics. These include non-embeddability into normed spaces results [KR06, AK07], lower bounds on sketching complexity [AK07], and sub-linear time algorithms [BEK+03]. Second, Ulam distance is no harder than edit distance over binary strings, at least up to constant approximation (see Theorem 1.2 from [AK07]). Thus, the Ulam metric is a specific roadblock that we must overcome before we may obtain improved results for general edit distance. Moreover, algorithms for Ulam metric have already found applications for a certain smoothed model for edit distance over binary strings [AK08]. We will discuss this application later.

In this paper, we give a near-tight bounds for estimating the Ulam distance up to a constant approximation, in sublinear time. Formally, given two non-repetitive strings A and B of length d over an alphabet Σ, with $|\Sigma| \geq d$, the problem is to output a constant approximation to $R = \mathrm{ed}(A, B)$. We show that $\tilde{\Theta}(d/R+\sqrt{d})$ time is sufficient and required for this problem.

THEOREM 1.1. (UPPER BOUND) *There exists a constant $\alpha > 1$ for which there exists a randomized algorithm that, given two non-repetitive strings $A, B \in \Sigma^d$, approximates $R = \mathrm{ed}(A, B)$ up to factor α in $\tilde{O}(d/R+\sqrt{d})$ time, with $2/3$ success probability.*

THEOREM 1.2. (LOWER BOUND) *For every constant $\alpha > 1$, if an algorithm approximates Ulam distance up to a factor α with $\geq 2/3$ success probability, then the algorithm must take $\Omega(d/R+\sqrt{d})$ time, where R is the edit distance between the two input strings. The lower bound holds for edit distance over binary strings as well.*

Our upper bound improves over the bound of $\tilde{O}(d/\sqrt{R})$ obtained in [AIK09]. We note that the bound from [AIK09] is tight in two extreme regimes: when $R \approx \Theta(d)$ and $R \approx \Theta(1)$. In contrast, our algorithm is tight in all the regimes of R, up to logarithmic factors. Our lower bound improves over the bound of $\Omega(d/R+\sqrt{R})$ that follows from [BEK+03] and folklore, giving a tighter (near-optimal) bound when $R = \Omega(\sqrt{d})$.

*Center for Computational Intractability at Princeton University (`andoni@mit.edu`). The work was done while the author was a student at MIT. Supported in part by NSF CAREER award CCR-0133849, David and Lucille Packard Fellowship and Alfred P. Sloan Fellowship .

†Princeton University (`hlnguyen@princeton.edu`). The work was done while the author was a student at MIT.

We further note that, in comparison, the best known upper bounds for general edit distance are currently much weaker: all sublinear time algorithms achieve a polynomial approximation only. Specifically, [BEK+03] can distinguish between $\mathrm{ed}(x,y) < d^{1-\epsilon}$ and $\mathrm{ed}(x,y) = \Omega(d)$ in $\tilde{O}(d^{1-2\epsilon})$ time. The algorithm of [AO09] can distinguish $\mathrm{ed}(x,y) < n^{\alpha}$ from $\mathrm{ed}(x,y) > n^{\beta}$ in $n^{\alpha+2(1-\beta)+o(1)}$ time.

Finally, an application of our upper bound theorem is a near-tight distance estimation algorithm for the smoothed edit distance model over binary strings defined in [AK08]. There, the authors provided a general reduction from distance estimation in the smoothed model of edit distance over binary strings to distance estimation of (worst-case) Ulam distance. (We will not define precisely the smoothed model of [AK08] as it will not appear further in the present article.)

Corollary 1.1. (Informal) *Let $x, y \in \{0,1\}^d$ be strings drawn from the smoothed model defined in [AK08]. Then, for every $\epsilon > 0$, we can compute* $\mathrm{ed}(x,y)$ *with* $O(1)$ *approximation in time* $\tilde{O}(d^{1+\epsilon}/\,\mathrm{ed}(x,y) + d^{0.5+\epsilon})$.

1.1 Overview of techniques We now describe the main ideas involved in proving Theorems 1.1 and 1.2.

Both the upper bound and lower bound exploit the fact that the Ulam metric is decomposable as a *sum-product* of Ulam metrics. The sum-product of Ulam metrics is a metric on k-tuples of non-repetitive strings, i.e., $(A_1, \dots A_k) \in (\Sigma^{\ell})^k$, with the distance between tuples $(A_1, \dots A_k)$ and $(B_1, \dots B_k)$ being defined as $\sum_{i=1}^{k} \mathrm{ed}(A_i, B_i)$. The resulting metric is a submetric of Ulam (over strings of length ℓk), as it can be realized by Ulam distance between two strings: $A' = A_1 \circ A_2 \circ \dots A_k$ and $B' = B_1 \circ B_2 \circ \dots B_k$, where for each coordinate i we relabel the symbols with new symbols from a fresh alphabet Σ_i, and $\circ$ is the concatenation operator.

Before presenting the ideas behind the upper bound, we rather start by presenting the ideas used for our lower bound, which is both simpler and instructive for presenting the ideas of the upper bound theorem. In the following, we will refer only to the *testing* problem, which asks to distinguish cases $\mathrm{ed}(A,B) < R$ versus $\mathrm{ed}(A,B) > \alpha R$, for some approximation factor α and fixed threshold $R > 1$. We note that considering algorithms for the testing problem is sufficient (and necessary) for both the upper and lower bound theorems.

Lower bound. The main question here is proving the bound of $\Omega(\sqrt{d})$, for $R > \sqrt{d}$, since the bound of $\Omega(d/R)$ follows immediately from the lower bound on testing Hamming distance. We first review the construction from [BEK+03], which gives a weaker bound, of $\Omega(\sqrt{R})$. Suppose the testing algorithm wants to distinguish the case $\mathrm{ed}(A,B) < R$ ("close pair") versus $\mathrm{ed}(A,B) > 2R$ ("far pair"). Then, the hard distribution generates A randomly from Σ^d, for $|\Sigma| \gg d$, and B is obtained from A by a cyclic rotation of A by an amount t chosen at random from either $[R]$ (for a "close pair") or $[100R]$ (for a "far pair"). Then, invoking a birthday paradox argument, one can show that the algorithm must sample $\Omega(\sqrt{R})$ positions in order to distinguish the two distributions.

To prove a sample lower bound of $\Omega(\sqrt{d})$, we consider the sum-product of $k \approx d/R$ copies of Ulam metric on strings of length R. To generate a "close pair" (A', B'), we pick $A' = (A_1, \dots A_k)$ randomly, and construct $B' = (B_1, \dots B_k)$ from A' where each B_i is a cyclic shift of A_i by an amount t_i chosen at random from $[R^2/d]$. To generate a "far pair", we do the same, except that for one position $i^* \in [d/R]$, we generate B_{i^*} from A_{i^*} via a cyclic rotation by a random amount $t_i \in [R]$. Now, for a coordinate i, to distinguish between a random shift $t_i \in [R^2/d]$ versus $t_i \in [R]$, by a birthday paradox argument, the algorithm needs to sample $\Omega(\sqrt{R^2/d})$ positions from A_i and B_i. Furthermore, since the algorithm does not know the value i^*, the algorithm has to sample $\Omega(\sqrt{R^2/d})$ positions for most of the coordinates $i \in [d/R]$. This gives a total bound of $\Omega(\sqrt{d})$ samples.

Upper bound. We are now ready to describe the ideas behind our upper bound. At a very high level, the upper bound does the converse of the lower bound. Suppose we want to test whether $\mathrm{ed}(A,B) < R$ or $\mathrm{ed}(A,B) > \alpha R$, where the approximation factor α is a large enough constant. First, we decompose the Ulam distance between input strings A, B into sum-product of $k = O(d/R)$ strings of length $\tilde{O}(R)$ by partitioning the strings $A = A_1 \circ \dots A_k$ and $B = B_1 \circ \dots B_k$ such that $\mathrm{ed}(A,B) = \sum_i \mathrm{ed}(A_i, B_i)$. Second, we design an algorithm for distinguishing whether the sum-product of Ulam distances $\sum_i \mathrm{ed}(A_i, B_i)$ is at most R or is bigger than αR. We reduce this step to the problem of *gap testing* the Ulam distance between two strings: given strings $u, v \in \{0,1\}^{\ell}$, distinguish whether $\mathrm{ed}(u,v) < a$ or $\mathrm{ed}(u,v) > b$ for $a \ll b$. In the third step, we design an algorithm for this gap-testing of Ulam distance, that runs in $\tilde{O}(\ell \cdot \sqrt{a}/b)$ time, where ℓ is the length of strings u, v. Each of these steps requires additional ideas, which we now briefly sketch.

We implement the first step, of reducing to testing of sum-product of $k = O(d/R)$ Ulam distances of strings of $\tilde{O}(R)$ length, as follows. Note that, in general, we cannot just directly partition A (and B) into blocks of equal length d/k since, in this case, $\sum_i \mathrm{ed}(A_i, B_i)$ can become as high as $k \cdot \mathrm{ed}(A,B)$. Instead, we proceed as follows. Consider the longest common substring of A and B, and let its positions in A and B be S_A and S_B respectively. We find some matching positions $a_1, \ldots a_{k-1} \in S_A$ and $b_1, \ldots b_{k-1} \in S_B$ such that $A[a_i] = B[b_i]$. Then, we partition the strings as $A = A[1, a_1 - 1] \circ A[a_1, a_2 - 1] \circ \ldots \circ A[a_{k-1}, d]$ and $B = B[1, b_1 - 1] \circ B[b_1, b_2 - 1] \circ \ldots \circ B[b_{k-1}, d]$. We show this can be done such that all the lengths of the substrings are $\tilde{O}(R)$, in $\tilde{O}(d/R + \sqrt{d})$ total time.

In the second step, we reduce testing sum-product of k Ulam metrics, $E = \sum_i \mathrm{ed}(A_i, B_i)$, to (many invocations of) the gap-testing problem of Ulam distance. The idea is to partition the coordinates $i \in [k]$ into levels corresponding to the contributing weight, and estimate separately the contribution of each level to E. Namely, we estimate c_j, the number of coordinates $i \in [k]$ such that $\mathrm{ed}(x_i, y_i) \geq 2^j$, for all $j = 0, \ldots \log R$. Then, $\sum_j c_j \cdot 2^j$ is a constant-factor approximation to E. For each j, we estimate c_j by subsampling coordinates i with rate $\approx 2^j/R$, and, for subsampled i's, testing whether $\mathrm{ed}(x_i, y_i) \geq 2^j$. So far, it looks like we did not save much: say, for $j = 1$, we subsample most of coordinates i for which we have to test whether $\mathrm{ed}(A_i, B_i) \geq R/2$. Naively, this would take at least $\Omega(\sqrt{R})$ time per coordinate, and $\Omega(d/\sqrt{R})$ for all coordinates. However, we note that, say, a big fraction of coordinates actually have distance $\mathrm{ed}(A_i, B_i) \leq O(1)$. Thus, at least for a fraction of i's, we need to only distinguish between $\mathrm{ed}(A_i, B_i) \leq O(1)$ and $\mathrm{ed}(A_i, B_i) \geq R/2$. Indeed, our gap-testing algorithm manages to do so in almost constant time.

More generally, the approach from above requires a gap-tester that can distinguish $\mathrm{ed}(x, y) < a$ versus $\mathrm{ed}(x, y) > b$ for all $a \ll b \leq R$. Our gap tester does so in time $\tilde{O}(\ell \cdot \frac{\sqrt{a}}{b})$, where ℓ is the length of the strings x and y. Note that, for the specific case of $b = O(a)$, our algorithm's performance recovers the performance of the algorithm from [AIK09]. To obtain our gap-testing algorithm, we develop an alternative characterization of Ulam distance, based on characterizations of [ACCL07, GJKK07].

In the end, when using our gap tester in the algorithm for testing sum-product of k Ulam distances of strings of length $\leq \ell$, we obtain a total time of $\tilde{O}(\frac{\ell}{R} \cdot (k + \sqrt{kR}))$. When $k = O(d/R)$ and $\ell = \tilde{O}(R)$ (as obtained in the first step), the running time becomes $\tilde{O}(d/R + \sqrt{d})$.

We proceed to describing our algorithms and the lower bound in detail.

2 Preliminaries and Notation

For a string A, let $A[i, j]$ denote the substring of A from position i to position j and, abusing notation, also the set of characters in that substring. If an index i is outside of the string $A \in \Sigma^l$, we extend, by convention, the string with extra symbols. Namely, for $\underline{i \leq 0}$, we let $A[i] = \underline{i}$, and, for $i > l$, we let $A[i] = \overline{(i - l)}$ (in particular the extension is the same for all strings). Then Σ will denote the extended alphabet. We assume all logs are in base 2. In the rest of the paper, we will make extensive use of the Chernoff bounds, which we recall below (see, e.g., [MR95]).

Fact 2.1. (Chernoff bound, [MR95]) *Let X_1, X_2,..., X_n be i.i.d. random variables and $p = \mathbb{E}[X_i]$, $\epsilon > 0$, $X_i \in \{0, 1\}$. Then, we have that*

- *If $\epsilon \leq 2e - 1$, then* $\Pr[|\sum_{i=1}^n X_i - pn| \geq \epsilon pn] \leq 2 \cdot e^{-\epsilon^2 pn/4}$,
- *If $\epsilon \geq 2e - 1$, then* $\Pr[|\sum_{i=1}^n X_i - pn| \geq \epsilon pn] \leq 2^{-(1+\epsilon)pn}$.

3 Distance Estimation for Ulam Distance

We now describe our algorithm for sublinear time distance estimation of Ulam distance, thus proving Theorem 1.1.

The main subroutine is for *testing* the Ulam distance between two strings. Namely, the tester has the following promise for input strings $A, B \in \Sigma^d$, and a given threshold $\log^5 d \leq R \leq d$:

- If $\mathrm{ed}(A, B) < \frac{R}{1400}$, then the tester returns CLOSE with probability at least 2/3.
- If $\mathrm{ed}(A, B) > R$ but $\mathrm{ed}(A, B) \leq 2R$, then the tester returns FAR with probability at least 2/3.

We note that such a tester is sufficient to approximate the distance $R^* = \mathrm{ed}(A, B)$. Indeed, we can run the tester for Ulam distance for each "guess" $R = d/2, d/4, d/8, \ldots$, and stop once the tester returns "FAR". More precisely, for each guess of R, we run the tester for Ulam distance for $O(\log d)$ times and take the majority answer. If the majority answer is "FAR", then we return the current value of R as an approximation to R^*.

Our tester for Ulam distance is described in Figure 1, and is named UlamTest(A, B, R). The tester works as follows. In step one, we decompose A and B into $k = O(d/R)$ substrings $A_1, \ldots, A_k$ and $B_1, \ldots, B_k$ such that the sum $\mathrm{ed}(A_1, B_1) + \ldots + \mathrm{ed}(A_k, B_k)$ equals $\mathrm{ed}(A, B)$. We refer to the distance between $(A_1, \ldots A_k)$ and $(B_1, \ldots B_k)$ as the sum-product of k copies of Ulam distance. In step two, the algorithm tests whether the sum of Ulam distances $\mathrm{ed}(A_1, B_1) + \ldots + \mathrm{ed}(A_k, B_k)$ is bigger than R or is smaller than $R/1400$. The first step is described below, and its main statement is Lemma 3.1. The second step is described in the next section, Section 4, and its main ingredient is Lemma 4.2. The two lemmas together imply Theorem 1.1.

Procedure UlamTest(A, B, R)
1. $a, b, m \leftarrow$ PartialAlign$(A, B, 2R)$
2. Return UlamProductTest$((A[1, a_1], B[1, b_1]), \ldots, (A[a_{d/2\beta R} + 1, d], B[b_{d/2\beta R} + 1, d]), R)$

Figure 1: The tester determining whether $\mathrm{ed}(A, B) > R$ or $\mathrm{ed}(A, B) < R/1400$. We assume that $\mathrm{ed}(A, B) \leq 2R$. Here, $\beta = C_1 \log^3 d$ for a sufficiently large constant $C_1 > 0$.

Procedure PartialAlign(A, B, R)
1. Split A and B into blocks of size βR.
2. $m_0 \leftarrow 0$
3. For $i \leftarrow 1$ to $\frac{d}{\beta R}$
4. For $j \leftarrow 4$ to $\log 4R$
5. Pick a random location p in $[(i-1) \cdot \beta R + 4R, i \cdot \beta R - 4R]$ of the i^{th} block. Pick $\gamma \cdot 2^{j/2}$ random positions from each of $A[p, p + 2^j]$ and $B[p + m_{i-1} - 2^j, p + m_{i-1} + 2 \cdot 2^j]$.
   If there is at least one collision $A[u] = B[v]$, then do the following. Choose any such collision u_i, v_i. Set $m_i \leftarrow v_i - u_i, a_i \leftarrow u_i, b_i \leftarrow v_i$. Stop the j loop and jump to the next i.
6. If the j-loop did not stop, then fail.
7. Return vectors a, b, and m.

Figure 2: Partial alignment of two strings. Here $\gamma = C_2 \log d$ for a sufficiently large constant $C_2 > 0$.

As described before, in the first step, we partition strings A and B as $A = A[1, a_1 - 1] \circ A[a_1, a_2 - 1] \circ \ldots \circ A[a_{k-1}, d]$ and $B = B[1, b_1 - 1] \circ B[b_1, b_2 - 1] \circ \ldots \circ B[b_{k-1}, d]$ for some $k = O(d/R)$, where a_i, b_i are such that positions a_i, b_i belong to the longest common subsequence of A and B respectively and $A[a_i] = B[b_i]$. In this case, it is immediate to note that $\mathrm{ed}(A, B) = \mathrm{ed}(A_1, B_1) + \ldots + \mathrm{ed}(A_k, B_k)$. While this may not always be possible, we show we can do it under the assumption that $\mathrm{ed}(A, B) \leq 2R$.

To be useful for the second step, we also need that $|a_{i+1} - a_i|, |b_{i+1} - b_i| \leq \tilde{O}(R)$ for all i. In the next subsection, we show how to find the positions a_i, b_i with the required properties.

3.1 Decomposition into a Sum Product of Ulam Distances

We now show how to find positions a_i, b_i, $i \in [k]$, belonging to some fixed longest common subsequence (LCS) of A and B and such that $|a_{i+1} - a_i|, |b_{i+1} - b_i| \leq \tilde{O}(R)$ for all i.

The main idea is as follows. Let S_A and S_B be the positions of the LCS in A and B respectively. First we partition the strings A into substrings of equal length βR, where $\beta = C_1 \log^3 d$ for large enough constant C_1. We consider each such substring $A[(i-1) \cdot \beta R + 1, i \cdot \beta R]$ and take the corresponding substring of B of length βR starting at s_i (where the notion "corresponding" will be clear momentarily; for the moment assume that $|s_i - (i-1) \cdot \beta R| \leq O(R)$). For each such pair of substrings $A[(i-1) \cdot \beta R + 1, i \cdot \beta R]$ and $B[s_i, s_i + \beta R - 1]$, we find a pair of positions a_i, b_i that belong to the sets S_A and S_B. We note that this is always possible since $\mathrm{ed}(A, B) \leq 2R$ and thus for each matching pair of positions a_i, b_i (in the LCS), we have that $|a_i - b_i| \leq 2R$. The notion of "corresponding substring" roughly means that we sequentially correct the start of the i^{th} substring of B according to the displacement obtained from the previous matching pair (a_{i-1}, b_{i-1}); i.e., $s_i = (i-1) \cdot \beta R + b_{i-1} - a_{i-1}$.

To find one such pair of positions (a_i, b_i), we employ random sampling from the two substrings and hope for a collision via the birthday paradox. In general, since the substrings may be at distance up to $O(R)$, we might need to sample roughly $\sqrt{R}$ positions, which proves to be too much (and gives a bound of $\tilde{O}(d/\sqrt{R})$ only).

Instead, the algorithm adapts to the *local distance* in the i^{th} pair of substrings of A and B. Thus, if the i^{th} pair of substrings are at distance f_i, then the algorithm will sample roughly $\sqrt{f_i}$ samples for this value of i (since, intuitively, the matching symbols differ in position by at most f_i, once we make the aforementioned correction to the start of the B's substring). This adaptation to the local distance between substrings is what gives us the improved bound: indeed, for every sequence f_i with $\sum f_i \leq O(R)$, we have that $\sum_{i=1}^{d/R} (O(1) + \sqrt{f_i}) = O(d/R + \sqrt{d})$.

The complete details of the algorithm are presented in Figure 2. We prove the following lemma.

Lemma 3.1. (PartialAlign) *Consider two non-repetitive strings A and B at distance $\mathrm{ed}(A, B) \leq R$ for some $R \in [d]$. Let S_A, S_B be the sets of indices of characters in A and B respectively of the longest common subsequence of A and B (note that $|S_A| = |S_B| \geq d - R$).*

Then, with probability at least 2/3, the following all hold. The vectors a and b returned by PartialAlign(A, B) *are subsets of S_A and S_B. Furthermore, $|a_{i+1} - a_i|, |b_{i+1} - b_i| \leq 2\beta R$ for all $i \in [d/\beta R]$. Finally, the running time of* PartialAlign(A, B)

is $\tilde{O}(d/R+\sqrt{d})$.

Proof. We prove that all a_i are from S_A with at least 0.9 probability. Since A and B are non-repetitive and b_i are such that $A[a_i]=B[b_i]$, then all b_i must be from S_B as well.

The proof is by induction on i. For convenience of notation, we set $a_0=b_0=0$. Now assume the inductive hypothesis: that all $a_k\in S_A$ for $k<i$. We prove that, conditioned on this event, the algorithm generates an $a_i\in S_A$ with probability at least $1-t_i$, where t_i is a function of a_{i-1} and will be defined later. We then prove that, conditioned on all $a_i\in S_A$, we have that $\sum t_i\le O(\gamma^2\log d/\beta)$, which will let us bound the failure probability.

Let f_i be the number of "bad positions" from the last match $A[a_{i-1}]=B[b_{i-1}]$ to the end of the i^{th} block in A and B. Formally, f_i is the number of positions in $A[a_{i-1}, i\cdot\beta R]\setminus S_A$ plus the number of positions in $B[b_{i-1}, m_{i-1}+i\cdot\beta R]\setminus S_B$.

The next claim bounds the probability that a "bad position" $s\notin S_A$ appears amongst the collisions for a fixed j, where a collision is a sampled pair (u,v) such that $A[u]=B[v]$.

Claim 3.1. *Fix some $j\le\log 4R$. The probability that a position $s\notin S_A$ appears in the set of collisions is at most $\frac{f_i\gamma^2}{\beta R}$.*

Proof. Note that we need to care only about symbols $s\in A[(i-1)\cdot\beta R+4R, i\cdot\beta R]\setminus S_A$. The probability of a fixed such symbol s yielding a collision is bounded by the probability that $s\in A[p,p+2^j]$, times the probability that s is sampled from $A[p,p+2^j]$ and $B[p+m_{i-1}, p+m_{i-1}+2^j]$, which is $\frac{2^j+1}{(\beta-8)R}\cdot\frac{\gamma 2^{j/2}}{2^j+1}\cdot\frac{\gamma 2^{j/2}}{3\cdot 2^j}\le\frac{\gamma^2}{\beta R}$. We now apply a union bound over all $s\in A[(i-1)\cdot\beta R+4R, i\cdot\beta R]\setminus S_A$ and use the fact that $|A[(i-1)\cdot\beta R+4R, i\cdot\beta R]\setminus S_A|\le f_i$, and thus obtain the desired conclusion.

The probability that $a_i\notin S_A$ is bounded by the probability that, for any $j\le\log 4R$, there exists a position $s\notin S_A$ that appears amongst the collisions. The latter probability is obtained by applying a union bound over all j to the bound from Claim 3.1, resulting in a bound of $\log 2R\cdot\frac{f_i\gamma^2}{\beta R}$.

We also need to bound the probability that a j-loop fails to stop. We bound this event using the claim from below, which specifies an upper bound on j at which the j-loop will stop.

Before stating and proving the claim, we introduce more notation. Consider fixed p and j. Let T_j be the set of symbols in $A[p,p+2^j]\setminus S_A$, and let $\bar{T}_j = A[p,p+2^j]\cap S_A$. Then $\mathbb{E}_p\left[|\bar{T}_j|\right]\le\frac{f_i}{(\beta-8)R}\cdot 2^{j+1}$. By Markov's inequality, with probability at least $1-\frac{20f_i}{(\beta-8)R}$, we have that $|\bar{T}_j|\le 0.1\cdot 2^j$.

Claim 3.2. *The j-loop stops for some j satisfying $2^j\le 2f_i$, with probability at least $\frac{21f_i}{(\beta-8)R}$. If $f_i=0$, then j-loop stops at $j=4$.*

Proof. Take the smallest j such that $2^j\ge f_i$. Condition on the event that $|\bar{T}_j|\le 0.1\cdot 2^j$. Then $|T_j|\ge 0.9\cdot 2^j$. Note that each $s\in T_j\subseteq A[p,p+2^j]$ also appears in $B[p+m_{i-1}-2^j, p+m_{i-1}+2\cdot 2^j]$ by definition of $f_i\le 2^j$.

Out of $\gamma 2^{j/2}$ samples in $A[p,p+2^j]$, at least $0.8\gamma\cdot 2^{j/2}$ are in T_j, with high probability (by usual Chernoff bound for $\gamma=\Omega(\log d)$). Let this set of samples belonging to T_j be denoted by W. Then, we can compute the probability that, out of the $\gamma 2^{j/2}$ sampled characters in $B[p+m_{i-1}-2^j, p+m_{i-1}+2\cdot 2^j]$, at least one is also in W: this probability is at least $1-(1-\frac{0.8\gamma 2^{j/2}}{3\cdot 2^j+1})^{\gamma 2^{j/2}}\ge 1-e^{-\Omega(\gamma^2)}\ge 1-d^{-\omega(1)}$.

Thus, we have at least one collision and the j-loop stops with probability at least $1-d^{-\omega(1)}-\frac{20f_i}{(\beta-8)R}\ge 1-\frac{21f_i}{(\beta-8)R}$.

We can now completely bound the probability that $a_i\in S_A$ and the algorithm finishes successfully the corresponding i^{th} step. Indeed, this probability is at least $1-\log 2R\cdot\frac{f_i\gamma^2}{\beta R}-\frac{21f_i}{(\beta-8)R}\ge 1-\frac{2\gamma^2\log R}{\beta}\cdot\frac{f_i}{R}$. We set $t_i=\frac{2\gamma^2\log R}{\beta}\cdot\frac{f_i}{R}$.

Finally, the probability that there exists some i for which $a_i\notin S_A$ is at most $\sum_i t_i\le\frac{4\gamma^2\log R}{\beta}\cdot\frac{\sum_i f_i}{R}$. We claim that $\sum_i f_i\le 4R$. Indeed, for fixed i, f_i is the number of positions in $A[a_{i-1}, i\cdot\beta R]\setminus S_A$ plus the number of positions in $B[b_{i-1}, m_{i-1}+i\cdot\beta R]\setminus S_B$ (conditioned on the fact that $a_{i-1}\in S_A$). In this case, each position $k\notin S_A$ contributes to f_i for at most 2 values of i (and same for $k\notin S_B$). Since also $|S_A|=|S_B|\ge d-R$, we have that $\sum_i f_i\le 4R$. Therefore, the probability that there exists some i for which $a_i\notin S_A$ is at most $16\gamma^2\log R/\beta<0.1$.

It remains to bound the running time. Assume that all $a_i\in S_A$. Using Claim 3.2, the running time of the algorithm is $\sum_{i=1}^{d/\beta R}O(1+\gamma\sqrt{f_i}\log d)=\tilde{O}(\frac{d}{\beta R}+\gamma\sqrt{\frac{d}{\beta R}\cdot R})=\tilde{O}(\sqrt{d}+\frac{d}{R})$, where we have applied the Cauchy-Schwartz the additional $O(\log d)$ appears because of the implementation of checking for collisions).

4 Tester for Sum-product of Ulam Distance

In this section, we describe a tester for a sum product of Ulam distances between tuples of strings. Given k pairs of strings $(A_1, B_1), \ldots, (A_k, B_k)$, where each string has length at most βR, for $\beta > 1$, and $\sum_{i=1}^k \mathrm{ed}(A_i, B_i) = O(R)$, the tester runs in $\tilde{O}(\beta(k + \sqrt{kR}))$ time and has the following promise:

- If $\sum_{i=1}^k \mathrm{ed}(A_i, B_i) < \frac{R}{1400}$, then the tester returns CLOSE with probability at least $\frac{2}{3}$.
- If $\sum_{i=1}^k \mathrm{ed}(A_i, B_i) > R$, then the tester returns FAR with probability at least $\frac{2}{3}$.

Procedure UlamProductTest$((A_1, B_1), \ldots, (A_k, B_k), R)$
1. for $i \leftarrow 0$ to $\log R$
2. $\quad\widehat{C}_i \leftarrow 0$
3. $\quad$Take a set S of pairs by picking each pair (A_u, B_u), $u \in [k]$, independently with probability $p_i = \min(6400 \frac{2^i \log^3(kR)}{R}, 1)$.
4. $\quad$For each $s \in S$
5. $\quad\quad$For $j \leftarrow 1$ to $i - 9$
6. $\quad\quad\quad$Run GapUlamTest$(A_s, B_s, 2^j, 2^i)$ for $O(\log(kR))$ times and take the majority answer. Stop the j loop if the majority answer is CLOSE.
7. $\quad\quad$If the j-loop is never stopped, increase $\widehat{C}_i$ by $\frac{1}{p_i} = \max(\frac{R}{6400 \cdot 2^i \log^3(kR)}, 1)$.
8. Compute the estimate $\widehat{d} = \sum_{i=0}^{\log R} 2^i \widehat{C}_i$.
9. If $\widehat{d} > 0.85R$, return FAR. Otherwise, return CLOSE.

Figure 3: Closeness tester for sum product of Ulam distance.

The tester UlamProductTest is described in details in Figure 3. The idea of the tester is to partition the pairs of strings into buckets of pairs of roughly equal distances and then approximate the number of pairs in each bucket. Specifically, let C_i be the number of indices u such that $\mathrm{ed}(A_u, B_u) \geq 2^i$. For each i, we compute an approximation of C_i with small additive and multiplicative errors. Finally, an approximation of $\sum_{u=1}^k \mathrm{ed}(A_u, B_u)$ can be obtained from the sum $\sum_{i=0}^{\log R} 2^i C_i$. A subroutine GapUlamTest(A, B, a, b) is used to differentiate between the case $\mathrm{ed}(A_u, B_u) < a$ and the case $\mathrm{ed}(A_u, B_u) > b$. Formally, GapUlamTest satisfies the following properties, which will be proved in a later section.

Lemma 4.1. (GapUlamTest) *Suppose we are given two non-repetitive strings A and B of length ℓ_A and ℓ_B, respectively, of characters in $[d]$ and two constants a, b satisfying $a \leq \frac{b}{512}$. Let $\ell = \max(\ell_A, \ell_B)$. With probability at least $2/3$,* GapUlamTest *runs in $\tilde{O}(\ell\sqrt{a}/b)$ time and distinguishes between the case $\mathrm{ed}(A, B) < a$ and the case $\mathrm{ed}(A, B) > b$.*

Assuming the properties of GapUlamTest, we now state the formal properties of UlamProductTest.

Lemma 4.2. (UlamProductTest) *Given k pairs of non-repetitive strings $(A_1, B_1), \ldots, (A_k, B_k)$ of characters in $[d]$ where the length of each string is bounded by βR and $\sum_{i=1}^k \mathrm{ed}(A_i, B_i) = O(R)$. With probability at least $2/3$,* UlamProductTest *runs in $\tilde{O}(\beta(k + \sqrt{kR}))$ time and correctly distinguishes between the cases $\sum_i \mathrm{ed}(A_i, B_i) > R$ and $\sum_i \mathrm{ed}(A_i, B_i) < R/1400$.*

Proof. Firstly, we prove the following approximation claim.

Claim 4.1. *Consider a set S of n elements and a subset T of m elements. Pick a random subset X of S by picking each element independently with probability p. Let $q = |X \cap T|$. Picking X can be implemented in expected $O(pn)$ time and $\Pr\left[|q/p - m| \geq 0.1m + \frac{6400 \log n}{p}\right] \leq \frac{1}{n^3}$.*

Proof. We pick X as follows. Divide S into blocks of size $\frac{1}{p}$ and use the binomial distribution to compute the number of samples in each block. Finally, pick the samples from each block according to the computed number of samples. The expected running time is $O(pn)$. Now consider two cases.

1. $m \leq \frac{6400 \log n}{p(2e-1)}$. By the Chernoff bound, $\Pr[|q/p - m| \geq m \cdot \frac{6400 \log n}{pm}] \leq 2^{-(1+\frac{6400 \log n}{mp})pm} \leq \frac{1}{n^3}$.
2. $m > \frac{6400 \log n}{p(2e-1)}$. By the Chernoff bound, $\Pr[|q/p - m| \geq 0.1m] \leq 2e^{-(0.1)^2 pm/4} \leq e^{\frac{16 \log n}{2e-1}} \leq \frac{1}{n^3}$.

This concludes the proof of claim 4.1.

By the Chernoff bound, the probability that the majority answer of $O(\log(kR))$ runs of GapUlamTest (on line 6 of UlamProductTest) is wrong, is bounded by $\frac{1}{k^3 R^3}$. The majority answer is taken $O(k \log^2(kR))$ times, so by the union bound, all majority answers from runs of GapUlamTest are correct with probability at least $1 - \frac{1}{kR}$. Thus, from now on, we assume all majority answers are correct.

Now we proceed to give upper and lower bounds on $\widehat{C}_i$, and hence, the distance estimate $\widehat{d}$. We consider the case $i < \log \frac{R}{6400 \cdot \log^3(kR)}$ (so $p_i < 1$). When i is large enough so that $p_i = 1$, the following bounds still hold because several estimation steps become exact computation.

We start by showing an upper bound for $\widehat{C}_i$ and $\widehat{d}$. Let N_i be the number of indices $s \in [k]$ such that $\mathrm{ed}(A_s, B_s) > \frac{2^i}{512}$. Let X_i be the number of indices $s \in S$ such that $\mathrm{ed}(A_s, B_s) > \frac{2^i}{512}$. By Claim 4.1, $\frac{1}{p_i} X_i \leq 1.1N_i + \frac{R}{2^i \log^2(kR)}$, w.h.p. Hence, $\widehat{C}_i \leq \frac{1}{p_i} X_i \leq 1.1N_i + \frac{R}{2^i \log^2(kR)}$ and $\widehat{d} \leq \sum_i 2^i(1.1N_i + \frac{R}{2^i \log^2(kR)}) < 1130 \sum_{i=1}^k \mathrm{ed}(A_i, B_i) + \frac{R}{\log(kR)}$.

Next we show a lower bound for $\widehat{C}_i$ and $\widehat{d}$. Let M_i be the number of indices $s \in [k]$ such that $\mathrm{ed}(A_s, B_s) > 2^i$. Let Y_i be the number of indices $s \in S$ such that $\mathrm{ed}(A_s, B_s) > 2^i$. By Claim 4.1, $\frac{1}{p_i} Y_i \geq 0.9M_i - \frac{R}{3 \cdot 2^i \log^2(kR)}$ w.h.p. Hence, $\widehat{C}_i \geq \frac{1}{p_i} Y_i \geq 0.9M_i - \frac{R}{2^i \log^2(kR)}$ and $\widehat{d} \geq \sum_i 2^i(0.9M_i - \frac{R}{2^i \log^2(kR)}) \geq 0.9 \cdot \sum_{i=1}^k \mathrm{ed}(A_i, B_i) - \frac{R}{\log(kR)}$.

Therefore, with probability at least 2/3, if $\sum_{i=1}^k \mathrm{ed}(A_i, B_i) > R$, then $\widehat{d} > 0.9R - R/\log(kR) > 0.85R$, and if $\sum_{i=1}^k \mathrm{ed}(A_i, B_i) \leq R/1400$, then $\widehat{d} < 1130R/1400 + R/\log kR < 0.85R$.

We now prove the stated running time of the algorithm. The expected number of times a fixed pair (A_u, B_u) is selected when we compute $\widehat{C}_i$ is $O(2^i \log^3(kR)/R)$. When a pair (A_u, B_u) is selected, the j-loop stops as soon as $2^j > \mathrm{ed}(A_u, B_u)$. Thus, the expected running time of the algorithm is

$$\sum_{i=0}^{\log R} \sum_{u=1}^{k} 2^i \log^3(kR)/R \cdot \tilde{O}\left(\frac{\beta R(\sqrt{\mathrm{ed}(A_u, B_u)} + 1)}{2^i}\right)$$
$$\leq \sum_{u=1}^{k} \tilde{O}(\beta(\sqrt{\mathrm{ed}(A_u, B_u)} + 1)) = \tilde{O}(\beta(k + \sqrt{kR})).$$

4.1 Gap Closeness Tester for Ulam distance In this section, we describe the details of GapUlamTest, a closeness tester differentiating between the case where the Ulam distance is very large and the case where the Ulam distance is very small. Specifically, we have two distance thresholds a and b satisfying $a < b/512$ and the algorithm should return FAR if the distance is at least b, and return $CLOSE$ if the distance is at most a.

The tester GapUlamTest is described in details in Figure 4. The idea of the algorithm is to divide both strings into small blocks and estimate the contribution of each block to the total distance. The contribution from each block comes from two sources: character movements within each block, and character movements between different blocks. Intuitively, the first kind of movements can be detected by character inversions within corresponding blocks in two strings. We approximate the number of movements of this kind by the number of characters witnessing a lot of inversions in their neighborhoods (similar to the characterizations from [ACCL07, GJKK07, AIK09]). The number of movements of the second kind is exactly the difference between the set of characters in the block in the first string and the set of characters in the corresponding block in the second string, which can be approximated by counting collisions between samples from two strings. Furthermore, since only an approximation of the sum of contributions from the blocks is needed, instead of computing the contributions from all blocks, we only sample some subset of blocks to estimate the sum. Specifically, for each i, we estimate n_i, the number of blocks contributing approximately 2^i or more. The total distance can be estimated by considering the sum $\sum_i n_i 2^i$. Intuitively, the larger i is, the finer the estimation of n_i we need, so the number of sampled blocks grows with i. On the other hand, the larger the distance 2^i, the easier it is to find mismatches between the corresponding blocks in two strings. It turns out these two effects cancel each other out and for each i, we can estimate the contributions from blocks contributing 2^i or more in $\tilde{O}(\frac{\ell\sqrt{a}}{b})$. Summing over all i, the total running time is $\tilde{O}(\frac{\ell\sqrt{a}}{b})$.

The algorithm uses the following characterization of the Ulam distance in order to approximate the two aforementioned forms of movement. We note that this lemma can be seen as a refinement of the characterizations from [ACCL07, GJKK07, AIK09].

LEMMA 4.3. *Consider two non-repetitive strings A and B. Let ℓ_A, ℓ_B be the length of A and B, respectively. W.l.o.g. assume $\ell_A \leq \ell_B$. Define $X = \sum_{k=0}^{\ell_A/a} |A[ka+1, (k+1)a] \setminus B[(k-1)a+1, (k+2)a]|$ i.e. the number of characters occurring in $A[ka+1, (k+1)a]$ but not in $B[(k-1)a+1, (k+2)a]$ for $k \in [\ell_A/a]$. Let $\delta \leq 1/2$ be a constant. Define Y_δ to be the number of pairs of indices u, v such that $A[u] = B[v]$, $A[u] \in A[ka+1, (k+1)a] \cap B[(k-1)a+1, (k+2)a]$ for some k and the symmetric difference $|A[u-t, u-1] \Delta B[v-t, v-1]| > 2\delta t$ for some $t \leq 4a$. Then*

- *If $\mathrm{ed}(A, B) \leq a$, then $X \leq a$ and $Y_\delta \leq 4a/\delta$.*
- *If $\mathrm{ed}(A, B) \geq b + \ell_B - \ell_A$, then $X + Y_\delta \geq \frac{b(1-\delta)}{2}$.*

Proof. A character is called red if it contributes to either X or Y_δ.

First, we show that if $\mathrm{ed}(A, B) \leq a$, then $X \leq a$ and $Y_\delta \leq 4a/\delta$. Let S_A, S_B be the set of indices of characters in A and B belonging to the longest common subsequence of A and B. Note that $|S_A| = |S_B| \geq \ell_B - a$. The characters in S_A cannot contribute to X, so $X \leq a$. Let T_δ be the set of all pairs of indices u, v such that $A[u] = B[v]$, and the symmetric

Procedure GapUlamTest(A, B, a, b)

1. Let ℓ_A, ℓ_B be the length of A and B. If $|\ell_B - \ell_A| > b/10$, return FAR.
2. Split A into ℓ_A/a blocks of size a.
3. For $i \leftarrow 0$ to $\log a - 1$
4. Set $\widehat{X}_i \leftarrow 0$
5. Pick a set S of blocks by picking each block $k \in [\ell_A/a]$ independently with probability $\min(O(\frac{2^i \log^3 \ell}{b}), 1)$.
6. For each sampled block $k \in S$
7. If $2^i \leq 6400\sqrt{a} \log \ell$,
8. Compute the number of characters in $A[ka+1,(k+1)a]$ that are not also contained in $B[(k-1)a+1,(k+2)a]$ (i.e., characters contributing to X) by reading the blocks entirely. Increase $\widehat{X}_i$ by 1 if this number of characters is at least 2^i.
9. If $2^i > 6400\sqrt{a} \log \ell$,
10. Read each character in $A[ka+1,(k+1)a]$ and $B[(k-1)a+1,(k+2)a]$ independently with probability $p = \min(O(\frac{\sqrt{a}\log \ell}{2^i}), 1)$ and let C be the number of collisions between the characters being read in A and in B. If $a - \frac{C}{p^2} > 0.9 \cdot 2^i$ then increase $\widehat{X}_i$ by 1.
11. Read each character in $A[ka+1,(k+1)a]$ and $B[(k-1)a+1,(k+2)a]$ independently with probability $r = \min(O(\frac{\log \ell}{2^{i/2}}), 1)$ and find the collisions. For each collision $A[u] = B[v]$, run YContributingTest(A, B, u, v, a). Let D be the number of characters for which the answer is CONTRIBUTING. If $\frac{D}{p^2} > 0.9 \cdot 2^i$, then increase $\widehat{Y}_i$ by 1.
12. Set $\widehat{X} \leftarrow \sum_i \frac{b}{\log^3 \ell} \widehat{X}_i$ and $\widehat{Y} \leftarrow \sum_i \frac{b}{\log^3 \ell} \widehat{Y}_i$
13. If $\widehat{X} + \widehat{Y} > \frac{b}{10}$, return FAR. Otherwise return CLOSE.

Figure 4: Tester distinguishing between the case $\mathrm{ed}(A, B) < a$ and the case $\mathrm{ed}(A, B) > b$.

difference $|A[u-t, u-1] \Delta B[v-t, v-1]| > 2\delta t$ for some $t \leq \ell_B$. Notice that $Y_\delta \leq |T_\delta|$. By [AIK09, Lemma 2.2], $|T_\delta| \leq 4a/\delta$.

Second, we show the contra-positive of the second assertion, i.e. if $X + Y_\delta < \frac{b(1-\delta)}{4}$ then $\mathrm{ed}(A, B) < b + \ell_B - \ell_A$. We select a common subsequence of A and B by the following removal procedure. For convenience, add two different special characters at the end of A and B and they are not red. Start from the end of A and go back until reaching the beginning of A. At position i, do the following. If $A[i-1]$ is not red, proceed to $i-1$. If $A[i-1]$ is red, let $j < i$ be the largest index where $A[j]$ is not red and $A[j]$ precedes $A[i]$ in B. Remove $A[j+1, i-1]$ and proceed to position j. The remaining string after the above process finishes is the common subsequence we need.

Now we bound the number of removed characters when we reach the ith position and $A[i]$ is not red. Because $A[i]$ is not red, $A[i] = B[u]$ for some u satisfying $|i - u| \leq 2a$. Consider two cases.

1. $i - j \leq 4a$. Because $A[i]$ is not red, $|A[j+1, i-1] \Delta B[u-i+j+1, u-1]| \leq 2\delta(i-j-1)$. All non-red characters in $A[j+1, i-1]$ contribute to the symmetric difference $|A[j+1, i-1] \Delta B[u-i+j+1, u-1]|$ so at least $1-\delta$ fraction of the deleted characters are red.

2. $i - j > 4a$. Because $A[i]$ is not red, $|A[i-4a+1, i-1] \Delta B[u-4a+1, u-1] \leq 8\delta a$. Thus, in $A[i-4a+1, i-1]$, at most $4\delta a$ characters are not red. We now show that all characters in $A[j+1, i-4a]$ are red. Indeed, any non-red character in $A[j+1, i-4a]$ must appear after the character $A[i] = B[u]$ in B by the definition of j, and thus it contributes to X and is red. Therefore, all characters in $A[j+1, i-4a]$ are red. In total, at least $4a - 4\delta a + (i - 4a - j) > (i-j-1)(1-\delta)$ red characters are removed.

In all cases, at least $1-\delta$ fraction of the deleted characters are red. Therefore, we get a common subsequence of A and B of length greater than $\ell_A - \frac{b}{2}$ so $\mathrm{ed}(A, B) < b + \ell_B - \ell_A$. This concludes the proof of Lemma 4.3.

The tester works by estimating the quantity X and $Y = Y_\delta$, for $\delta = 1/2$ in the above lemma. To detect characters contributing to Y_δ, we use the YContributingTest described in Figure 5. The following lemma shows that YContributingTest correctly tests if a character contributes to Y.

Lemma 4.4. (YContributingTest) *With probability at least $1 - \frac{1}{\ell^2}$, if $|A[u-z, u-1] \Delta B[v-z, v-1]| > z$ for some $z < 4a$, then* YContributingTest *returns CONTRIBUTING, and if $|A[u-z, u-1] \Delta B[v-z, v-1]| < 0.8z$ for all $z < 4a$, then* YContributingTest *returns NOT-CONTRIBUTING. The expected running time of* YContributingTest *is $\tilde{O}(\sqrt{a})$.*

Proof. Let $N_t = |A[u-1.01^t, u-1] \cap B[v-1.01^t, v-1]|$. Let D_t be the number of collisions between samples from $|A[u-1.01^t, u-1]$ and $B[v-1.01^t, v-1]|$. We have

$\mathbb{E}\left[D_t/q^2\right] = N_t$ and $\mathrm{Var}[D_t/q^2] = (1-q^2)N_t/q^2$. By the Chebyshev inequality, $\Pr[|D_t/q^2 - N_t| \geq 0.05 \cdot 1.01^t] \leq \frac{400(1-q^2)N_t}{q^2 1.01^{2t}} < \frac{1}{10}$. Consider two cases.

1. $|A[u-z,u-1]\Delta B[v-z,v-1]| > z$ for some $z < 4a$. Choose $t = \lfloor \log_{1.01} z \rfloor$. Then, $|A[u-1.01^t, u-1]\Delta B[v-1.01^t, v-1]| > 0.98 \cdot 1.1^t$. Therefore, $N < 0.51 \cdot 1.1^t$. By the Chernoff bound, with probability at least $1 - \frac{1}{\ell^2}$, for the majority of the times, the number of collisions is at most $0.55 \cdot 1.01^t$ and the algorithm returns CONTRIBUTING.

2. $|A[u-z,u-1]\Delta B[v-z,v-1]| < 0.8z$ for all $z < 4a$. Therefore, $N > 0.6 \cdot 1.01^t$. By the Chernoff bound, with probability at least $1 - \frac{1}{\ell^2}$, the number of collisions is at least $0.55 \cdot 1.01^t$ the majority of the times for all t and the algorithm returns NOT-CONTRIBUTING.

The expected number of characters being read by YContributingTest, and hence, the expected running time, is $\tilde{O}(\sqrt{a})$. This concludes the proof of Lemma 4.4.

Procedure YContributingTest(A, B, u, v, a)
1. For $t \leftarrow 0$ to $\log_{1.1} 4a$
2. Repeat the following $O(\log \ell)$ times.
3. Read each character in $A[u-1.01^t, u-1]$ and $B[v-1.01^t, v-1]$ independently with probability $q = O(\frac{1}{\sqrt{1.01^t}})$ and count the number of collisions.
4. If the number of collisions is at most $0.55 \cdot 1.01^t$ the majority of the times then return CONTRIBUTING.
5. If CONTRIBUTING is never returned then return NOT-CONTRIBUTING.

Figure 5: A procedure for checking if $|A[u-t, u-1]\Delta B[v-t, v-1]|$ is large for some $t < 4a$.

Now we proceed to proving Lemma 4.1.

Proof. We first show $\widehat{X}$ approximates X. Consider a sampled block $A[ka+1, (k+1)a]$. Let $M_k = |A[ka+1,(k+1)a] \setminus B[(k-1)a+1,(k+2)a]|$ and $N_k = |A[ka+1,(k+1)a] \cap B[(k-1)a+1,(k+2)a]|$. Note that $M_k + N_k = a$. When $2^i \leq 6400\sqrt{a}\log \ell$, we get the exact value of M_k and N_k by reading the whole block. Now consider the case $2^i > 6400\sqrt{a}\log\ell$. Let C_k be the number of collisions between samples from $A[ka+1,(k+1)a]$ and samples from $B[(k-1)a+1,(k+2)a]$ when reading each symbol with probability p. We have $\mathbb{E}\left[C_k\right] = p^2 N_k$. Consider two cases.

1. $N_k > \frac{0.1 \cdot 2^i}{2e-1}$. By the Chernoff bound, $\Pr[|C_k/p^2 - N_k| \geq N_k \frac{0.1\cdot 2^i}{N_k}] \leq 2e^{-(0.1\cdot 2^i/N_k)^2 p^2 N_k/4} < \frac{1}{\ell^2}$.

2. $N_k \leq \frac{0.1\cdot 2^i}{2e-1}$. By the Chernoff bound, $\Pr[|C_k/p^2 - N_k| \geq N_k \frac{0.1\cdot 2^i}{N_k}] \leq 2^{-(1+0.1\cdot 2^i/N_k)p^2 N_k} < \frac{1}{\ell^2}$.

Thus, with high probability, $|\frac{C_k}{p^2} - N_k| < 0.1 \cdot 2^i$. Therefore, the test on line 10 passes if $M_k \geq 2^i$ and fails if $M_k < 0.8 \cdot 2^i$. Let H_i be the number of indices $k \in [\ell/a]$ such that $|A[ka+1,(k+1)a] \setminus B[(k-1)a+1,(k+2)a]| \geq 2^i$ and K_i be the number of indices $k \in [\ell/a]$ such that $|A[ka+1,(k+1)a] \setminus B[(k-1)a+1,(k+2)a]| > 0.8 \cdot 2^i$.

By Claim 4.1, with high probability, $\frac{b}{2^i \log^3 \ell}\widehat{X}_i < 1.1K_i + \frac{b}{2^i \log^2 \ell}$ and thus, $\widehat{X} < \frac{1.1\cdot 2}{0.8}X + \frac{b}{\log \ell}$. Similarly, with high probability, $\frac{b}{2^i \log^3 \ell}\widehat{X}_i > 0.9H_i - \frac{b}{2^i \log^2 \ell}$ and thus, $\widehat{X} > 0.9X - \frac{b}{\log \ell}$.

We now show $\widehat{Y}$ approximates Y. Let $P_{k,t}$ be the number of characters $A[u] = B[v]$ such that $ka+1 \leq u \leq (k+1)a$ and $(k-1)a+1 \leq v \leq (k+2)a$, and $|A[u-z,u-1]\Delta B[v-z,v-1]| > 2tz$ for some $z < 4a$. By Lemma 4.4 and the union bound, with probability at least $1 - \frac{1}{\ell}$, all YContributingTest calls give correct answers. Each character contributing to Y is picked in both A and B on line 11 with probability $r^2 = O(\frac{\log^2 \ell}{2^i})$. By Claim 4.1, with high probability, $D < 1.1 \cdot P_{k,0.4} + \frac{2^i}{\log \ell}$ and $D > 0.9 \cdot P_{k,0.5} - \frac{2^i}{\log \ell}$.

Let P_i be the number of indices $k \in [\ell_A/a]$ such that $P_{k,0.5} > 2^i$ and Q_i be the number of indices $k \in [\ell_A/a]$ such that $P_{k,0.4} > 0.8 \cdot 2^i$. By Claim 4.1, with high probability, $0.9 \cdot P_i - \frac{b}{2^i \log^2 \ell} < \frac{b}{2^i \log^3 \ell}Y_i < 1.1Q_i + \frac{b}{2^i \log^2 \ell}$. Thus, $0.9 \cdot Y_{0.5} - \frac{b}{\log \ell} < \widehat{Y} < 2.5Y_{0.4} + \frac{b}{\log \ell}$.

Therefore, with high probability, if $\mathrm{ed}(A,B) < a$, $\widehat{X} + \widehat{Y} < 3a + \frac{2.5\cdot 4}{0.4}a + \frac{2b}{\log \ell} < b/10$ and if $\mathrm{ed}(A,B) > b$ and $|\ell_B - \ell_A| < b/10$, $\widehat{X} + \widehat{Y} > \frac{0.8b(1-0.5)}{2} \geq b/10$. Therefore, the algorithm answers correctly.

The expected running time of the algorithm is

$$\sum_{i=0}^{\log a} \frac{\ell 2^i \log^3 \ell}{ab} \tilde{O}\left(\min(a, \tfrac{a\sqrt{a}}{2^i}) + \frac{a\sqrt{a}}{2^i}\right) =$$

$$\tilde{O}\left(\frac{\ell\sqrt{a}}{b}\right).$$

5 Lower bound: Proof of Theorem 1.2

To prove Theorem 1.2, we follow the outline given in the techniques section. The lower bound $\Omega(\frac{d}{R})$ follows directly from the folklore lower bound on testing the Hamming distance, so we concentrate on the regime $2\sqrt{d} \leq R \leq d/4\alpha$.

Let $\ell = 4\alpha R$ and $r = R^2/d$. We define two distributions over pairs of permutations of $[\ell]$. Let μ_f be the distribution over pairs (A, B) where A is a random permutation of $[\ell]$ and B is obtained from A by a cyclic rotation of A, moving t_f characters at the end of A to the beginning with t_f drawn uniformly from $[\ell/4, \ell/2]$. Let μ_c be the distribution over pairs (A, B) where A is a random permutation and B is obtained from A by a cyclic rotation of A by an amount t_c drawn uniformly from $[r/4, r/2]$.

By an argument similar to [BEK+03, Theorem 3], there is a constant c such that any deterministic algorithm that, with probability at least 5/9, distinguishes a pair (A, B) drawn from μ_f from a pair (A, B) drawn from μ_c, must make at least $c\sqrt{r}$ queries. For completeness, below we include the sketch of the proof of the claim.

LEMMA 5.1. ([BEK+03]) *For any deterministic algorithm M making at most $\sqrt{r}/5$ queries,*

$$\left| \Pr_{(A,B)\leftarrow\mu_f}[M(A,B)=1] - \Pr_{(A,B)\leftarrow\mu_c}[M(A,B)=1] \right| < 1/9.$$

Proof. [Proof sketch.] Let B be a cyclic rotation of A by an amount $t \le \ell/2$. The longest common subsequence of A and B has length exactly $\ell - t$. Thus, when (A, B) is drawn from μ_f, $\mathrm{ed}(A, B) \ge t_f \ge \ell/4$. When (A, B) is drawn from μ_c, $\mathrm{ed}(A, B) \le 2t_c \le r$. Define R_c to be the event that the input to M is drawn from μ_c and M queries some position i in A and position $i + t_c$ in B for some $i \in [\ell]$. Also define R_f to be event that the input to M is drawn from μ_f and M queries i in A and $i + t_f$ in B for some $i \in [\ell]$. When R_c and R_f does not happen, all queried characters are distinct and random so M can not distinguish between μ_c and μ_f. Thus,

$$\left| \Pr_{(A,B)\leftarrow\mu_f}[M(A,B)=1] - \Pr_{(A,B)\leftarrow\mu_c}[M(A,B)=1] \right| \le \max(\Pr[R_c], \Pr[R_f]).$$

When M finds two identical characters, it can correctly distinguish between μ_f and μ_c. When M has not seen two identical characters in A and B, all queried characters are random and distinct. Therefore, adaptivity does not help and we can assume M makes all queries at once. After q queries on A and B, at most $(q/2)^2$ shifts are checked and because t_c and t_f are chosen uniformly at random, we have

$$\max(\Pr[R_c], \Pr[R_f]) \le \max\left(\frac{(q/2)^2}{r/4}, \frac{(q/2)^2}{\ell/4}\right) < 1/9.$$

This concludes the proof of Lemma 5.1.

We now proceed to proving Theorem 1.2. Assume for contradiction that there is an algorithm M' that with probability at least 2/3, takes at most $\frac{\sqrt{d}}{216\alpha}$ queries and approximates the Ulam distance between two input strings A, B up to a constant factor α. One can construct an algorithm to distinguish μ_c and μ_f with at most $\sqrt{r}/5$ samples as follows. Given some pair (A, B) from either μ_c or μ_f, construct a new pair (P, Q), which consists of d/ℓ blocks each, where one block at a random index $k \in [d/\ell]$ is (A, B) and all the others are drawn i.i.d. from μ_c. Run M' on (P, Q). If M' queries the block (A, B) at least $\frac{R}{5\sqrt{d}}$ times, then the algorithm aborts. Now, our output is "FAR" iff either

1. M' outputs "FAR", or,
2. M' takes at least $\frac{R}{5\sqrt{d}}$ from the block (A, B) (i.e., the algorithm aborts).

Clearly, our algorithm makes at most $\frac{R}{5\sqrt{d}}$ queries to (A, B).

Now, we prove the correctness of the algorithm. When (A, B) is drawn from μ_c, $\mathrm{ed}(P, Q) \le dr/\ell = \frac{R}{4\alpha}$. When (A, B) is drawn from μ_f, $\mathrm{ed}(P, Q) \ge \ell/4 = \alpha R$. Because $\frac{R}{4\alpha} \cdot \alpha < R$, if aborting is ignored, M should output the correct answer with probability at least 2/3. If $(A, B) \in \mu_f$, then with probability at least 2/3, the output would be "FAR" (at least by criterion 1). Now consider the case $(A, B) \in \mu_c$. With probability at least 2/3, criterion 1 cannot happen. Let N_i be the number of queries M makes on the i^{th} block. All blocks are drawn i.i.d. from μ_c regardless of the value of k and k is not revealed to M so $\mathbb{E}[N_i] = \mathbb{E}[N_i | k = 1] = \ldots = \mathbb{E}[N_i | k = d/\ell]\ \forall i$. Thus, $\mathbb{E}[N_k] = \frac{\mathbb{E}\left[\sum_{i=1}^{d/\ell} N_i\right]}{d/\ell} \le \frac{R}{54\sqrt{d}}$. By the Markov inequality, with probability at least 8/9, $N_k \le \frac{R}{6\sqrt{d}}$ so criterion 2 cannot happen, either. Therefore, with probability at least 5/9, the output when $(A, B) \in \mu_c$ would be "CLOSE".

The resulting algorithm distinguishes μ_c from μ_f which contradicts Lemma 5.1.

The claim for edit distance on binary strings follows immediately using Theorem 1.2 of [AK07].

References

[ACCL07] N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31:371–383, 2007. Previously appeared in RANDOM'04.

[AIK09] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Overcoming the ℓ_1 non-embeddability barrier: Algorithms for product metrics. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 865–874, 2009.

[AJKS02] Miklós Ajtai, T. S. Jayram, Ravi Kumar, and D. Sivakumar. Approximate counting of inversions in a data stream. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 370–379, 2002.

[AK07] Alexandr Andoni and Robert Krauthgamer. The computational hardness of estimating edit distance. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 724–734, 2007. Accepted to *SIAM Journal on Computing* (FOCS'07 special issue).

[AK08] Alexandr Andoni and Robert Krauthgamer. The smoothed complexity of edit distance. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 357–369, 2008.

[AO09] Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 199–204, 2009.

[BEK+03] Tuğkan Batu, Funda Ergün, Joe Kilian, Avner Magen, Sofya Raskhodnikova, Ronitt Rubinfeld, and Rahul Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 316–324, 2003.

[GJKK07] Parikshit Gopalan, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Estimating the sortedness of a data stream. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 318–327, 2007.

[KR06] Robert Krauthgamer and Yuval Rabani. Improved lower bounds for embeddings into l_1. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1010–1017, 2006.

[Mar95] John I. Marden. *Analyzing and Modeling Rank Data*. Monographs on Statistics and Applied Probability 64. CRC Press, 1995.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.