Accurate and Nearly Optimal Sublinear Approximations to Ulam Distance*

Timothy Naumovitz† Michael Saks‡ C. Seshadhri§

Abstract

The *Ulam distance* between two permutations of length n is the minimum number of insertions and deletions needed to transform one sequence into the other. Equivalently, the Ulam distance d is n minus the length of the longest common subsequence (LCS) between the permutations. Our main result is an algorithm, that for any fixed $\varepsilon > 0$, provides a $(1+\varepsilon)$-multiplicative approximation for d in $\tilde{O}_\varepsilon(n/d + \sqrt{n})$ time, which has been shown to be optimal up to polylogarithmic factors. This is the first sublinear time algorithm (provided that $d = (\log n)^{\omega(1)}$) that obtains arbitrarily good multiplicative approximations to the Ulam distance. The previous best bound is an $O(1)$-approximation (with a large constant) by Andoni and Nguyen (2010) with the same running time bound (ignoring polylogarithmic factors).

The improvement in the approximation factor from $O(1)$ to $(1+\varepsilon)$ allows for significantly more powerful sublinear algorithms. For example, for any fixed $\delta > 0$, we can get additive δn approximations for the LCS between permutations in $\tilde{O}_\delta(\sqrt{n})$ time. Previous sublinear algorithms require δ to be at least $1-1/C$, where C is the approximation factor, which is close to 1 when C is large.

Our algorithm is obtained by abstracting the basic algorithmic framework of Andoni and Nguyen, and combining it with the sublinear approximations for the longest increasing subsequence by Saks and Seshadhri (2010).

1 Introduction

Computing the Longest Common Subsequence (LCS) between two strings x and y is a fundamental algorithmic problem with numerous applications. Except for some polylogarithmic improvements, the best known algorithm is still the textbook quadratic time dynamic program. Recent results show that strongly subquadratic algorithms for LCS would violate the Strong Exponential Time Hypothesis [1, 7, 8].

A notable special case of LCS is when the strings x and y have no repeated characters. Equivalently, it is convenient to think of both x and y as permutations of length n. In this case, a simple reduction to the longest increasing subsequence yields an $O(n \log n)$ algorithm. We give the first sublinear time algorithm that gives arbitrarily good approximations to the LCS length.

THEOREM 1.1. *Assume random access to two strings x and y of length n that have no repeated characters. Fix any $\delta > 0$. There is an algorithm that, with probability at least $2/3$, outputs an additive δn estimate to the LCS length in time $\widetilde{O}_\delta(\sqrt{n})$ time.*

More generally, our techniques give multiplicative approximations to the Ulam distance between x and y. This is the minimum number of insertions or deletions required to transform x to y. Alternately, the Ulam distance is n minus the LCS length [10, 3]. (We note that some later results allow substitution operations as well [9, 5].) These authors were concerned with constant factor approximations, and this distinction was unimportant for them.)

THEOREM 1.2. *Assume random access to two strings x and y of length n that have no repeated characters. Fix any $\varepsilon > 0$ and let d be n minus the LCS length. There is an algorithm that outputs, with probability at least $2/3$, a $(1+\varepsilon)$-approximation to d in time $\widetilde{O}_\varepsilon(n/d + \sqrt{n})$.*

1.1 Connections to Previous Work The previous best result for estimating d, as defined in Theorem 1.2, is the result of Andoni-Nguyen [5] (henceforth AN). They get a (large) constant factor approximation with the same running time as Theorem 1.2. Moreover, they prove that $\tilde{O}(n/d + \sqrt{n})$ is optimal. These bounds were improvements over previous results [6, 4]. We note that the improvement of the constant to $1+\varepsilon$ is essential for getting Theorem 1.1. Roughly speaking, if one can get a c-approximation to the Ulam distance, that implies (at best) an additive $(1-1/c)n$ approximation to the LCS.

A related result is that of Saks-Seshadhri (henceforth SS), on sublinear time algorithms to approximate the length of the Longest Increasing Subsequence (LIS) of an array [16]. This problem is (almost) equivalent to approximating the LCS length between an arbitrary input permutation and the identity permutation. In the language of Theorem 1.2, they give a $(1+\varepsilon)$-approximation to the Ulam distance d in time $\widetilde{O}_\varepsilon(\text{poly}(n/d))$. At a high level, our algorithm is a com-

*Supported by NSF grant CCF-1218711 and by Simons Foundation Award 332622.

†Department of Mathematics, Rutgers University. tnaumovi@math.rutgers.edu

‡Department of Mathematics, Rutgers University. saks@math.rutgers.edu

§Department of Computer Science, University of California, Santa Cruz. sesh@ucsc.edu

bination of (suitably modified and improved versions of) AN and SS.

2 Preliminaries

2.1 Sequences, boxes, LCS and loss functions

Natural number intervals, sequences and subsequences: $\mathbb{N}$ denotes the set of nonnegative integers and $\mathbb{N}^+$ is the set of positive integers. Interval notation $[a, b]$, $(a, b]$, etc. is used both for intervals of nonnegative integers and real numbers; the meaning will be clear from context. For a positive integer r, $[r]$ denotes the set $\{1, ...r\}$. A natural number sequence of length r is represented as a function $f : [r] \longrightarrow \mathbb{N}$. The sequence is *nonrepeating* if the function is 1-1. A subsequence of f is specified by a subset $I \subseteq [r]$ and is denoted $f(I)$. Writing I as $\{i_1, \ldots, i_s\}$ of $[r]$, with $i_1 < \cdots < i_s$, we think of $f(I)$ as the subsequence $f(i_1), \ldots, f(i_s)$. When I is an interval $[a, b]$ or $(a, b]$, $f(I)$ is a *consecutive subsequence*.

Pairs of sequences: Throughout this paper, f_X and f_Y denote nonrepeating sequences of lengths n_X and n_Y, respectively, referred to as the *horizontal sequence* and the *vertical sequence*. These are the input to the longest common subsequence problem. An element of the domain of f_X is an *X-index* (horizontal index), and an element of the domain of f_Y is a *Y-index* (vertical index).

X-intervals, Y-intervals, boxes, height and width: A subinterval I_X of $[n_X]$ is an *X-interval* and a subinterval I_Y of $[n_Y]$ is a *Y-interval*. The product of an X-interval and a Y-interval is a *box* $\mathcal{B} = I_X \times I_Y$; in particular the box $[n_X] \times [n_Y]$ is denoted $\mathcal{U}$.

For $P \in \mathcal{U}$, we write $x(P)$ and $y(P)$, respectively, for its x and y coordinates. For a box $\mathcal{B}$, $X(\mathcal{B})$ and $Y(\mathcal{B})$ denote the X-interval and Y-interval such that $\mathcal{B} = X(\mathcal{B}) \times Y(\mathcal{B})$. A box $\mathcal{B}$ has *width* $w(B) = |X(\mathcal{B})|$ and *height* $h(B) = |Y(\mathcal{B})|$. For simplicity, we also use $d_{min}(\mathcal{B}) = \min(w(\mathcal{B}), h(\mathcal{B}))$ and $d_{max}(\mathcal{B}) = \max(w(\mathcal{B}), h(\mathcal{B}))$.

Matches and the relations $P \sim Q$ and $P \nsim Q$: A pair $(x, y) \in \mathcal{U}$ is a *match* if $f_X(x) = f_Y(y)$. For matches $P, Q \in \mathcal{U}$, $P < Q$ means $x(P) < x(Q)$ and $y(P) < y(Q)$. If either $P < Q$ or $Q < P$, then P is *comparable* to Q, denoted $P \sim Q$. If neither hold, then P is a *violation* with Q, denoted $P \nsim Q$.

Common subsequence and the function `lcs`: A sequence of matches $M_1, M_2, \ldots, M_k$ satisfying $M_1 < \cdots < M_k$ is called a *common subsequence* of f_X and f_Y. Define $\texttt{lcs}_{f_X, f_Y}$ to be the length of the longest common subsequence of f_X and f_Y. If all points of a common subsequence lie in a box $\mathcal{B}$, this is a common subsequence in $\mathcal{B}$. Define $\texttt{lcs}(\mathcal{B})$ to be the length of the longest common subsequence in $\mathcal{B}$. $\texttt{lcs}(\mathcal{B})$ has the following geometric interpretation. The set of all matches gives a subset of points in the upper quadrant, and $\texttt{lcs}(\mathcal{B})$ is the longest sequence of points that are increasing (i.e., go up and to the right).

Loss function: A *loss function* assigns to each box a nonnegative integer that measures the number of indices that do not participate in the LCS. We define four loss functions.

- $d_{in}(\mathcal{B}) = |\mathcal{P}(\mathcal{B})| - \texttt{lcs}(\mathcal{B})$. $d_{in}(\mathcal{B})$ measures the number of input points in $\mathcal{B}$ which do not lie on a fixed LCS of $\mathcal{B}$.
- $\texttt{Xloss}(\mathcal{B}) = w(\mathcal{B}) - \texttt{lcs}(\mathcal{B})$. $\texttt{Xloss}(\mathcal{B})$ measures the number of X-indices of $\mathcal{B}$ that are not matched by a fixed LCS of $\mathcal{B}$.
- $\texttt{Yloss}(\mathcal{B}) = h(\mathcal{B}) - \texttt{lcs}(\mathcal{B})$. $\texttt{Yloss}(\mathcal{B})$ measures the number of Y-indices of $\mathcal{B}$ that are not matched by a fixed LCS of $\mathcal{B}$. It turns out that we will not need to use the loss function $\texttt{Yloss}(\mathcal{B})$ directly, but it is convenient to define, and we will use a modified version of it, which will be introduced in section 9.1.
- $\texttt{uloss}(\mathcal{B}) = w(\mathcal{B}) + h(\mathcal{B}) - 2 \cdot \texttt{lcs}(\mathcal{B})$. Alternatively, $\texttt{uloss}(\mathcal{B}) = \texttt{Xloss}(\mathcal{B}) + \texttt{Yloss}(\mathcal{B})$. $\texttt{uloss}(\mathcal{B})$ measures the number of indices (X or Y) of $\mathcal{B}$ that are not matched by a fixed LCS of $\mathcal{B}$. Note that $\texttt{uloss}(\mathcal{B}) = 2 \cdot \texttt{Xloss}(\mathcal{B}) + h(\mathcal{B}) - w(\mathcal{B})$.

Box sequences and box chains: A *box sequence* is a list of boxes such that each successive box is entirely to the right of the previous. We use the notation $\vec{\mathcal{B}}$ to denote a box sequence. We write $\mathcal{B} \in \vec{\mathcal{B}}$ to mean that the box $\mathcal{B}$ appears in $\vec{\mathcal{B}}$. A *box chain* $\vec{\mathcal{T}}$ is a box sequence $\vec{\mathcal{T}} = (\mathcal{T}_1, \cdots, \mathcal{T}_k)$ such that (for all i), if P_i is the top right corner of $\mathcal{T}_i$ and Q_{i+1} is the bottom left corner of $\mathcal{T}_{i+1}$, then $P_i < Q_{i+1}$. We say that a box chain $\vec{\mathcal{T}}$ *spans* $\mathcal{B}$ if $\mathcal{B}$ is the smallest box containing $\vec{\mathcal{T}}$.

2.2 Tail Bounds

We recall a version of the Chernoff-Hoeffding bound from [13] for binomial random variables:

THEOREM 2.1. (Chernoff-Hoeffding) *Let $X \sim Bin(n, p)$. Then if $\mu = np$, $\delta \in (0, 1)$,*

- $\Pr[X < (1 - \delta)\mu] \leq e^{-\delta^2 \mu/2}$.
- $\Pr[X > (1 + \delta)\mu] \leq e^{-\delta^2 \mu/3}$.

We also make use of the following version from [14]:

THEOREM 2.2. (Chernoff Bound) *Let $X \sim Bin(n, p)$. Then for $\delta > 0$,*

- *If $\delta \leq 2e - 1$, then* $\Pr[|X - pn| \geq \delta pn] \leq 2e^{-\delta^2 pn/4}$
- *If $\delta \geq 2e - 1$, then* $\Pr[|X - pn| \geq \delta pn] \leq 2^{-(1+\delta)pn}$.

2.3 Parameter approximation and Gap tests

Fix a real valued parameter P. A real number $\tilde{P}$ is

a (τ, δ)-approximation to P if $|\tilde{P} - P| \leq \tau P + \delta$. If $\tilde{P}$ is obtained by a randomized algorithm, then $\tilde{P}$ is a (τ, δ)-approximation with failure at most κ if $\Pr[|\tilde{P} - P| > \tau P + \delta] \leq \kappa$.

A (τ, δ)-*gap test* for P takes as input two parameters, the *lower threshold* a and the *upper threshold* $b \geq (1+\tau)a$ and outputs SMALL or BIG. The gap test is said to fail if either $P \leq a - \delta$ and it outputs BIG or $P > b$ and it outputs SMALL. We say that the gap test operates with failure probability at most κ if on any input (a, b) with $b \geq (1+\tau)a$ the probability of failure is at most κ. When $\delta = 0$, we simply call the test a τ-gap test. We refer to the pair (τ, δ) (or the parameter τ when $\delta = 0$) as the *quality* of the gap test. The following proposition is easily proven by a geometric search over parameters to a gap test.

PROPOSITION 2.1. *Consider P whose range is $\{0, \ldots, r\}$. Suppose there exists a (τ, δ)-gap test for P that runs in time T with failure probability at most κ. Then, there exists a $(2\tau + \tau^2, \delta)$-approximation algorithm for P that runs in time at most $T(\log_{1+\tau}(r) + 2)$ with failure at most $\kappa(\log_{1+\tau}(r) + 2)$.*

Proof. Construct an approximation algorithm **ALG** as follows. Run the (τ, δ)-gap test for successive values of j starting from 0, where $a_j = r/(1+\tau)^{j+1}$ and $b_j = r/(1+\tau)^j$. The first time the test says BIG output a_j as the estimate. If j reaches the value $t = \log_{1+\tau} r + 1$ and the test returns SMALL with parameters a_t and b_t, output 0 as the estimate.

ALG consists of at most $(\log_{1+\tau}(r) + 2)$ calls to the gap test, so the running time of **ALG** is within the stated bound. Additionally, by a union bound, the probability that none of these calls fail is at least $1 - \kappa(\log_{1+\tau}(r) + 2)$. Therefore, we will assume that none of the gap test calls fail.

First, suppose $P = 0$. In this case, the gap test is guaranteed to return small whenever the value of a_j is greater than δ. As a result, the output of **ALG** will either be some value of a_j which is at most δ, or 0 if every gap test call returns SMALL. In either case, the output of **ALG** differs from the value of P by at most δ.

Suppose $P > 0$. Since $b_t < 1$, the gap test will return BIG for some value of j. Let k be the smallest value of j such that the gap test returns BIG. The output of **ALG** is a_k. By the guarantee of the gap test, the value of P must be at least $a_k - \delta$, which is within an additive δ of the output of **ALG**. If $k > 0$, then the gap test returned SMALL with parameters a_{k-1} and b_{k-1}. Therefore, the value of P must be at most b_{k-1}. Since $b_{k-1} = (1+\tau)^2 a_k$, the value of P is at most $(1 + 2\tau + \tau^2)$ times the output of **ALG**, so these two quantities differ by at most a $(2\tau + \tau^2)$ multiplicative factor. If instead $k = 0$, then since the value of P is at most $r = b_k$, the same bound holds. □

3 Overview of the algorithm

Our results use ideas as well as the high level structure of the AN algorithm [5]. This algorithm runs in time $\tilde{O}(\frac{n}{d} + \sqrt{n})$, and with high probability, provides a $(C, 0)$-approximation to the Ulam distance for some large C. We carefully analyze AN to identify the places that contribute significantly to the multiplicative loss. Along the way we abstract and simplify the key ideas of the algorithm.

Prop. 2.1 allows us to focus on gap tests. We also need gap tests for the Ulam distance within boxes. The AN algorithm can be viewed as constructing a hierarchy of three gap tests, each with successively better run times.

- A slow gap test whose run time is $\tilde{O}(\frac{w(\mathcal{B})^{3/2}}{b})$.
- A medium gap test whose run time is $\tilde{O}(\frac{w(\mathcal{B})\sqrt{a}}{b})$.
- A fast gap test whose run time is $\tilde{O}(\frac{w(\mathcal{B})}{b} + \sqrt{a})$.

The slow gap test is built using inversion/violation counting techniques, common to property testing algorithms for LIS [11, 15, 2]. The medium gap test is built from the slow gap test and the fast gap test is built from the medium gap test. (Both are done through related, but different techniques.) We refer to the methods that do this as the *speed-up procedures*. The drawback of their algorithm is the multiplicative approximation guarantee (quality) of the gap test. The slow gap test has quality 36, the medium gap test has quality 512, and the fast gap test has quality about 1400.

Our improvement has two components:

- **Higher quality speed-up:** By a careful analysis and modification of the AN framework, we avoid the explosion of the quality factor from the slow test to the medium, and from the medium to the fast. Thus, the fast gap test has quality only slightly worse than the slow gap test.
- **Higher quality slow gap test:** The main bottleneck in the quality comes from the slow test. Their slow test relies on a standard technique from monotonicity testing called *inversion counting* and this method cannot yield approximation factors better than 2. The SS algorithm [16] overcomes this limitation to get a polylogarithmic time algorithm for approximating distance to monotonicity with an additive error δn.

We adapt the SS algorithm to get a slow gap test with quality $(\varepsilon, 0)$. The SS algorithm is an incredibly complex beast. Our main insight is that a gap test can be constructed by a careful modification to SS that avoids getting into the intricacies of SS.

We describe some of the key ideas behind these two components.

3.1 Getting the speed-up routines Our speed-up routines are directly inspired by those in the AN algorithm, but differ in several respects: firstly, we provide a conceptual simplification of AN for a cleaner exposition; secondly, the requirement of a higher quality speed-up introduces new technical ideas.

Switching between loss functions: We begin with the latter. Routine calculations give tighter bounds for many of the AN arguments. But the transformation from the slow gap test to the medium test (at the very best) takes a β-gap test and produces a $(\beta+1)$-gap test. (Note that the latter gives a $(1+\beta, 0)$-approximation.) This suffices for a constant factor approximation to the Ulam distance, but cannot yield an (overall) $(\varepsilon, 0)$-approximation. Avoiding this requires getting into the subtlety of defining losses.

All the AN gap tests (and proofs) deal with $\texttt{Xloss}(\mathcal{B}) = w(\mathcal{B}) - \texttt{lcs}(\mathcal{B})$. Note that this is potentially smaller that the Ulam distance $\texttt{uloss}(\mathcal{B})$, which is $\texttt{Xloss}(\mathcal{B})+h(\mathcal{B})-\texttt{lcs}(\mathcal{B}) \geq \texttt{Xloss}(\mathcal{B})$. Thus, the AN gap tests are dealing with a more stringent loss function. Our slow gap test also yields an $\texttt{Xloss}(\cdot)$ bound, but our medium gap test only yields a guarantee for $\texttt{uloss}(\mathcal{B})$. Of course, this suffices for the final algorithm, but it is crucial for avoiding the quality loss from a β-gap test to a $(\beta+1)$-gap test.

Loss indicators: One key conceptual simplification we introduce in building the medium and fast gap test is an *indicator for a parameter* P. Suppose we are trying to estimate a nonnegative parameter P, which has an upper bound r. A (β_0, β_1)-*quality* P-*indicator* is a 0-1 valued random variable Z_P, satisfying $\Pr[Z_P = 1] \in [\frac{P}{r} - \beta_0, \frac{P}{r} + \beta_1]$. In particular Z_P is an estimator for P having bias at most $\max(\beta_0, \beta_1)$. We will consider indicators for $\texttt{uloss}(\mathcal{B})$ and $\texttt{Xloss}(\mathcal{B})$, which we refer to as *loss indicators*. Here we take the trivial upper bound for $\texttt{uloss}(\mathcal{B})$ to be $w(\mathcal{B}) + h(\mathcal{B})$ and for $\texttt{Xloss}(\mathcal{B})$ to be $w(\mathcal{B})$.

Given such an indicator, we can easily construct a gap test for P with thresholds a and b, where $b > a + (\beta_0 + \beta_1)r$. We define the procedure **GTfromLI**, which converts a loss indicator L into a gap test. The inputs to **GTfromLI** are an approximation parameter $\tau > 0$ (which will determine the quality of the gap test), an error parameter $\kappa > 0$ (which will be the probability of error of the gap test), and lower and upper thresholds a and b. The loss indicator L is a subroutine that is passed into **GTfromLI**. Our construction is simply to take $O(\frac{(1+\tau)r}{b\tau^3}\log(\frac{1}{\kappa}))$ samples of the estimator L and declare SMALL iff the fraction of 1's is less than $(1-\frac{\tau}{3})(\frac{b}{r} - \beta_0)$, where r is the upper bound for L and (β_0, β_1) is the quality of L. **GTfromLI**$(a, b, \tau, \kappa; L)$ is the procedure given by this construction. We state the following proposition which shows that **GTfromLI** is indeed a gap test, and specifies the quality, error, and running time parameters of the gap test.

PROPOSITION 3.1. *Suppose L is a loss indicator for a nonnegative parameter P with upper bound r and quality (β_0, β_1). Then for $\tau, \kappa > 0$,* **GTfromLI**$(a, b, \tau, \kappa; L)$ *is a $(\tau, (\beta_0 + \beta_1)r)$-gap test for P with error at most κ which, for any a, b with $a \geq (\beta_0 + \beta_1)r$, $b > (1+\tau)a$, runs in time $O(\frac{(1+\tau)r}{b\tau^3}\log(\frac{1}{\kappa}))$ times the running time of L.*

Proof. Suppose the probability that L outputs 1 is q. If we take m samples of L, the sum of the outputs is a random variable $Z \sim Bin(m, q)$. By Theorem 2.1, for $\delta > 0$, $\Pr[Z < (1-\delta)mq] \leq e^{-\delta^2 mq/2}$. Therefore, if $m \geq \frac{2\log(\frac{1}{\kappa})}{\delta^2 q}$, this is at most κ. Similarly, if $m \geq \frac{3\log(\frac{1}{\kappa})}{\delta^2 q}$, then $\Pr[Z > (1+\delta)mq]$ is at most κ.

For parameters a, b, **GTfromLI**$(a, b, \tau, \kappa; L)$ takes m samples of L and declares SMALL iff the fraction of 1's is less than $(1-\frac{\tau}{3})(\frac{b}{r} - \beta_0)$. If the value of P is at least b, then $q \geq \frac{b}{r} - \beta_0$. Therefore, taking $\delta = \frac{\tau}{3}$, if $m \geq \frac{18\log(\frac{1}{\kappa})}{\tau^2(\frac{b}{r}-\beta_0)}$, the probability that we declare SMALL is at most κ.

If on the other hand, the value of P is at most $a - (\beta_0 + \beta_1)r$, then $q \leq a - \beta_0 r \leq \frac{b - \beta_0 r}{1+\tau}$. Therefore, taking $\delta = \frac{\tau}{3}$, if $m \geq \frac{27\log(\frac{1}{\kappa})}{\tau^2(\frac{b}{r}-\beta_0)}$, the probability that we declare BIG is at most κ.

Suppose $a \geq (\beta_0 + \beta_1)r$, $b > (1+\tau)a$. Then $\beta_0 \leq \frac{b}{(1+\tau)r}$, so $\frac{b}{r} - \beta_0 \geq \frac{\tau}{1+\tau}\frac{b}{r}$. Thus, in either of the two cases above, taking $m = O(\frac{(1+\tau)r}{b\tau^3}\log(\frac{1}{\kappa}))$ is sufficient to bound the probability of error by κ, establishing the claim. □

The above procedure constructs a $(\tau, (\beta_0 + \beta_1)r)$-gap test for P, however we will often want to obtain a purely multiplicative gap test for P. It turns out that if β_0, β_1 are (approximate) multiples of the parameter P, then we can treat these additive terms as multiplicative terms, meaning that the gap test we have for P can be stated as a purely multiplicative gap test.

PROPOSITION 3.2. *Suppose G is a $(\tau, \varepsilon(P+1))$-gap test for an integer parameter P with error κ. Then for input thresholds a, b with $a \geq 1$, G is a $(\tau + 2\varepsilon)$-gap test for P with error κ.*

Proof. Let a, b be input thresholds with $b \geq (1+\tau+2\varepsilon)a$, $a \geq 1$. We know that G is a $(\tau, \varepsilon(P+1))$-gap test

for P, so if we have input parameters a', b' with $b' \geq (1+\tau)a'$, the test will return BIG if the value of P is at least b' and SMALL if the value of P is at most $a' - \varepsilon(P+1)$. Set $b' = b$, $a' = \frac{b}{1+\tau}$. If the value of P is at least b, the test will return BIG. If the value of P is at most $a \leq a' - 2\varepsilon b$, then if P is at least 1, $a' - 2\varepsilon b \leq a' - \varepsilon(2P) \leq a' - \varepsilon(P+1)$, so the test returns SMALL. If P is 0, then since $a \geq 1 = P+1$ and $2b \geq a$, $a' - 2\varepsilon b \leq a' - \varepsilon a \leq a' - \varepsilon(P+1)$, so the test again returns SMALL. □

It turns out that we can frame most of AN in this language, reformulating it as algorithms that generate indicators for various parameters and run in very fast expected time. This allows for a cleaner description of AN, and also eases the way to plug in better slow gap tests.

3.2 Getting a better slow gap test Our improved slow gap test is obtained by adapting the fast LIS (longest increasing subsequence) algorithm of SS. In the LIS problem, the input is a single sequence f and the problem is to find the longest monotonically increasing subsequence of f. We can represent the problem geometrically in the plane by the set of points $P(x) = (x, f(x))$ for $x \in [n]$. Just as with the LCS problem, the LIS problem is to find the longest increasing chain of points. Let $\texttt{loss}_f$ denote the number of points not on the LIS. The SS algorithm obtains an approximation having error $\varepsilon\texttt{loss}_f$ that runs in time $\widetilde{O}((\frac{w(\mathcal{B})}{\texttt{loss}_f})^{O(1)})$. We want to adapt the SS algorithm to get the base gap test. The first step in the adaptation is to improve the running time of the SS algorithm to $\widetilde{O}(\frac{w(\mathcal{B})}{\texttt{loss}_f})$. We sketch that improvement at the end of this section.

Even though LCS and LIS have the same geometric representation, we cannot adapt the LIS algorithm directly to LCS. The reason is that the LIS algorithm has the ability to query the point $P(x)$ associated to a particular X-index x at unit cost. The LCS algorithm can't do this because, given x, the algorithm can read the symbol $f_X(x)$, but doesn't know which Y-index it's matched to in f_Y.

Examining the SS algorithm, one sees that the key primitive that it needs is, given a box of width w and a time bound $\widetilde{O}(t)$, in time $\widetilde{O}(t)$ either generate a random point in the box, or determine that there are at most $\frac{w}{t}$ points in the box. This primitive is implemented in the LIS algorithm by sampling $\widetilde{O}(t)$ random indices and returning the first point in the box or if none, saying that there are at most $\frac{w}{t}$ points. So to adapt this algorithm to LCS, we just need to simulate that primitive.

To simulate this primitive, we select $\widetilde{O}(t\sqrt{w(\mathcal{B})})$ random X-indices from $X(\mathcal{B})$ and $\widetilde{O}(t\sqrt{w(\mathcal{B})})$ random Y-indices from $Y(\mathcal{B})$ and select a random match from among them if there is a match. Provided that $h(\mathcal{B})$ and $w(\mathcal{B})$ are not too far apart, a "birthday paradox" calculation shows that if the number of points is at least $\frac{w(\mathcal{B})}{t}$, then this procedure will almost certainly find a match. Thus, we are able to simulate the primitive at a multiplicative cost of $\widetilde{O}(\sqrt{w(\mathcal{B})})$. This simulation fails on unbalanced boxes, but it turns out that these boxes are unimportant for getting an accurate estimate. Therefore the overall running time of the resulting LCS algorithm is $\widetilde{O}(\frac{w(\mathcal{B})}{\texttt{Xloss}(\mathcal{B})}\sqrt{w(\mathcal{B})})$.

Now we describe our improvement to the LIS algorithm. While the run time bound given by [16] for a multiplicative $(1+\varepsilon)$ approximation to LIS distance is $\widetilde{O_\varepsilon}(\text{poly}(w(\mathcal{B})/\texttt{Xloss}(\mathcal{B})))$, [16] also proves a $\widetilde{O_\delta}(w(\mathcal{B})/\texttt{Xloss}(\mathcal{B}))$ bound for approximating LIS distance to within an additive δn term. We observe that though the full LIS algorithm of [16] is iterative with potentially many levels of recursion, when $\texttt{Xloss}(\mathcal{B})$ is small, most runs of the algorithm end in just one level. At a high level, we can combine a version of SS that runs for only one level of recursion with the additive δn version, to get an overall improved running time. This requires some careful parameter settings, and yields an $(\varepsilon, 0)$-approximation algorithm for $\texttt{Xloss}(\mathcal{B})$ that runs in $\widetilde{O_\varepsilon}(w(\mathcal{B})/\texttt{Xloss}(\mathcal{B}))$ time.

3.3 The high level structure The construction consists of nine algorithms:

- $\mathbf{XLI}_0$ is a point classification procedure such that, when run on a uniformly random X-index of the input box $\mathcal{T}$, is an $\texttt{Xloss}$-indicator for $\mathcal{T}$ running in time $\widetilde{O}_{\varepsilon_3}(\sqrt{w(\mathcal{T})})$.
- $\mathbf{BGT}$ is the slow gap test for $\texttt{Xloss}(\mathcal{T})$, running in time $\tilde{O}(\frac{w(\mathcal{T})}{b}\sqrt{w(\mathcal{T})})$, using $\mathbf{XLI}_0$ as a subroutine.
- $\mathbf{XLI}_1$ generates an $\texttt{Xloss}$-indicator for a box $\mathcal{T}$ in time $\tilde{O}(\sqrt{w(\mathcal{T})})$, using $\mathbf{BGT}$ as a subroutine.
- $\mathbf{XLI}_2$ generates an $\texttt{Xloss}$-indicator for a box $\mathcal{B}$ in time $\tilde{O}(\sqrt{r})$ by running $\mathbf{XLI}_1$ on a box of width r.
- $\mathbf{UGT}_1$ is the medium gap test for $\texttt{uloss}(\mathcal{B})$, using $\mathbf{XLI}_2$ as a subroutine.
- $\mathbf{ULI}_1$ generates a $\texttt{uloss}$-indicator for a box $\mathcal{B}$ that runs in time $\tilde{O}(\sqrt{\texttt{uloss}(\mathcal{B})}+1)$, using $\mathbf{UGT}_1$ as a subroutine.
- $\mathbf{ULI}_2$ generates a $\texttt{uloss}$-indicator for a box $\mathcal{U}$ that runs in time $\tilde{O}(\frac{b}{\sqrt{n}}+1)$ by running $\mathbf{ULI}_1$ on a subbox of width $\tilde{O}(b)$.
- $\mathbf{UGT}_2$ is the fast gap test for $\texttt{uloss}(\mathcal{B})$, using $\mathbf{ULI}_2$ as a subroutine.

• **Main** is the final approximation algorithm for $\texttt{uloss}(\mathcal{U})$, using $\mathbf{UGT}_2$ as a subroutine.

The reader should note that the medium gap test for $\texttt{uloss}(\mathcal{B})$ uses an $\texttt{Xloss}$-indicator rather than a $\texttt{uloss}$-indicator. This is one of the technical subtleties of the algorithm and the proof.

The main lemmas are stated in section 8. Each lemma establishes guarantees for one of these procedures given the guarantees of the previous procedure. Some of these lemmas are proven more easily than others. Many of the proofs share similarities in structure with each other. More specifically, the proofs that build a loss indicator from a gap test all share some common characteristics with each other, and the proofs that build a gap test from a loss indicator likewise share some common characteristics with each other. In the next two sections, we will outline these proofs, focusing mainly on these commonalities.

4 Constructing loss indicators from gap tests

We sketch the construction and proof for the loss indicator procedures $\mathbf{XLI}_1$ and $\mathbf{ULI}_1$ (Lemma 8.3 and Lemma 8.6). The full proofs are in section 9. In both procedures, the loss indicator is built from a gap test. The procedures $\mathbf{XLI}_2$ and $\mathbf{ULI}_2$ are loss indicators constructed from loss indicators as opposed to gap tests; we will discuss them in the next section along with the procedures $\mathbf{UGT}_1$ and $\mathbf{UGT}_2$. $\mathbf{XLI}_2$ and $\mathbf{UGT}_1$ together form a process which constructs a gap test from the loss indicator given by $\mathbf{XLI}_1$; $\mathbf{XLI}_2$ both improves the running time of the indicator (so that $\mathbf{UGT}_1$ will be more efficient than **BGT**) and alters the quality of the indicator in a way that makes it compatible with the analysis of $\mathbf{UGT}_1$. $\mathbf{ULI}_2$ and $\mathbf{UGT}_2$ have a similar relationship. For these reasons, we choose to group the exposition of these four procedures together in the next section as opposed to grouping any of them with $\mathbf{XLI}_1$ and $\mathbf{ULI}_1$ in this section.

The constructions of $\mathbf{XLI}_1$ and $\mathbf{ULI}_1$ are elementary and generic; they are applicable to general parameter indicators and not just loss indicators. For convenience, we will assume that all randomized procedures make no error. (In general, we can ensure this with high probability and take union bounds.)

We begin with the setting of Lemma 8.3, but simplify parameters for this discussion. Fix a box $\mathcal{B}$. Let $\tau > 0$ be an arbitrarily small, but fixed constant. We have a procedure **BGT** for $\texttt{Xloss}(\mathcal{B})$ that runs in time $\widetilde{O}((w(\mathcal{B})/b)\sqrt{w(\mathcal{B})})$. This means that given arbitrary $a, \delta > 0$ and $b > (1+\tau)a$, **BGT** correctly determines if $\texttt{Xloss}(\mathcal{B}) \geq b$ (BIG) or $\texttt{Xloss}(\mathcal{B}) \leq a - \delta$ (SMALL). More abstractly, we have a (τ, δ)-gap test for the parameter $P = \texttt{Xloss}(\mathcal{B})$, with a maximum value $w(\mathcal{B})$. We wish to construct a loss indicator for P with quality $(\Theta(\tau P/w(\mathcal{B})), \Theta((\tau P + \delta)/w(\mathcal{B})))$.

A direct approach would be to simply estimate P using the gap test, and finally flip an appropriate biased coin. Define $c_i = w(\mathcal{B})/(1+\tau)^i$. For increasing values of i, we can run the gap test with $a = c_i$ and $b = c_{i-1}$, and stop when the test reports BIG. As shown by Prop. 2.1, the corresponding value of a is a $(2\tau + \tau^2, \delta)$-approximation to P. The run time of a gap test is inversely proportional to b, and thus the very last test dominates the run time of the indicator. The last value of b is $\Theta(P) = \Theta(\texttt{Xloss}(\mathcal{B}))$. Thus, the run time of the indicator is $\widetilde{O}((w(\mathcal{B})/\texttt{Xloss}(\mathcal{B}))\sqrt{w(\mathcal{B})})$.

Inspired by the analysis in [5], we can design a faster indicator whose run time is only $\widetilde{O}(\sqrt{w(\mathcal{B})})$. The procedure $\mathbf{XLI}_1$ is constructed using this indicator.

We use $G_{i,j}$ (for $i > j$) to denote the gap test with $a = c_i$ and $b = c_j$. The simple indicator described above runs gap test $G_{1,0}$, $G_{2,1}$, $G_{3,2}$, ... until one of them returns BIG. The indicator described below uses a probabilistic stopping rule to get a better running time.

1. Initialize h to 0.
2. While $c_h > 1$
   (a) Perform $G_{h+1,h}$.
   (b) If gap test returns BIG, output 1.
   (c) Else (gap test returned SMALL):
      i. With probability $1 - 1/(1+\tau)$, output 0.
      ii. (With remaining prob.) Increment h.

For simplicity of exposition, let us suppose that $P = c_k = w(\mathcal{B})/(1+\tau)^k$. The probability that $G_{h+1,h}$ is run is exactly $(1+\tau)^{-h}$. The only way the indicator outputs 1 is when $G_{k+1,k}$ is run, which happens with probability $(1+\tau)^{-k} = P/w(\mathcal{B})$. Thus, the indicator has the right expectation. The expected running time is $\widetilde{O}(\sum_{h \leq \log_{1+\tau} w(\mathcal{B})} (1+\tau)^{-h}(w(\mathcal{B})/c_h)\sqrt{w(\mathcal{B})})$. Plugging in $c_h = w(\mathcal{B})/(1+\tau)^h$, the expected running time is $\widetilde{O}(\sqrt{w(\mathcal{B})})$, as desired.

To design $\mathbf{ULI}_1$, we require an indicator that is even faster. Suppose we have a gap test for a parameter P that runs in time $\widetilde{O}((w(\mathcal{B})/b)\sqrt{a})$. Then, we can get an indicator with an expected running time of $\widetilde{O}(\sqrt{P})$, significantly better than $\widetilde{O}(\sqrt{w(\mathcal{B})})$. (This is precisely the statement of Lemma 8.6.)

The key is to run tests of the form $G_{i,j}$ where $i > j + 1$. Depending on whether it outputs BIG or SMALL, we can modify the i, j to make the gap smaller. The description of the indicator follows.

1. Initialize i to $\lceil \log_{1+\tau} w(\mathcal{B}) \rceil$ (maximum possible index value) and j to 0
2. While indicator is not assigned 0 or 1

(a) Perform $G_{i,j}$.
(b) If test returns BIG then:
  i. If $i = j + 1$, output 1.
  ii. Else ($i > j + 1$): decrement i.
(c) Else (test returns SMALL):
  i. If $i = j + 1$, output 0.
  ii. With prob. $1 - 1/(1+\tau)$, output 0. With remaining prob., increment j.

It is not hard to show that this procedure outputs 1 with the correct probability (The details of the proof appear in section 9). The only probabilistic operation is the increment of j (just as in the previous indicator), and the analysis is almost identical.

The running time analysis crucially uses the improved gap test run time of $\widetilde{O}((w(\mathcal{B})/b)\sqrt{a})$. Observe that when $b = \Theta(w(\mathcal{B}))$, the gap test runs in time $\widetilde{O}(\sqrt{a})$. The choice of a in all the gap tests above is never larger than P. The observation together with the run time analysis of the previous indicator yields an expected run time of $\widetilde{O}(\sqrt{P})$.

5 Constructing gap tests from loss indicators

Our method for constructing a gap test from a loss indicator involves first constructing an improved loss indicator using the input loss indicator, and then converting this improved loss indicator to a gap test. In each case, we break this process into two procedures implementing these two steps. We now sketch the construction and correctness proof for the construction of the improved loss indicators $\mathbf{XLI}_2$ and $\mathbf{ULI}_2$ and gap test procedures $\mathbf{UGT}_1$ and $\mathbf{UGT}_2$. The full proofs are in section 9.

In both procedures, a gap test with thresholds a, b is built from a loss indicator. As mentioned in section 3.1, we can naively construct a gap test from a loss indicator by simply running the loss indicator on the input box a sufficient number of times and using the average value of the loss indicator as an estimate for the value of the parameter. However, this process would not yield any speed-up from the previous gap test because the loss indicator run on the input box is too expensive. So instead, we construct a loss indicator which consists of choosing a random subbox of width $O(b)$ (according to a carefully chosen distribution) and applying the initial loss indicator to that subbox. This new loss indicator is much faster and can be used to build a gap test (via Prop. 3.1) which runs in the desired time.

In the case of $\mathbf{UGT}_2$, the approach is straightforward (Again, the full details of the proofs appear in section 9). Part of the AN algorithm is a procedure **PartialAlign**, which constructs a box chain $\mathcal{T}_1, \cdots \mathcal{T}_M$ for a box $\mathcal{B}$ with $M = \tilde{O}(\frac{n}{b})$ boxes each of width $\tilde{O}(b)$ such that if $\texttt{uloss}(\mathcal{B}) < b$, then with high probability the ulam distance of $\mathcal{B}$ is the sum of the ulam distances of the boxes in the box chain. The running time of **PartialAlign** is $\tilde{O}(\frac{n}{b} + \sqrt{n})$, which is within the cost we're aiming at for our gap test. Let I_j be a $\texttt{uloss}$-indicator for $\mathcal{T}_j$. The procedure $\mathbf{ULI}_2$ constructs a $\texttt{uloss}$-indicator for $\mathcal{B}$ by simply selecting one of the boxes in the chain so that $\mathcal{T}_j$ is selected with probability $\frac{w(\mathcal{T}_j)}{w(\mathcal{B})}$. It is easy to check that $\mathbf{ULI}_2$ outputs a $\texttt{uloss}$-indicator for $\mathcal{B}$ whose expected running time is the (weighted) average running time of I_j. The running time of I_j is $\tilde{O}(\sqrt{\texttt{uloss}(\mathcal{T}_j)} + 1)$. By concavity, the weighted average (which is the running time of $\mathbf{ULI}_2$) can be bounded by $\tilde{O}(\sqrt{\frac{b \cdot \texttt{uloss}(\mathcal{B})}{n}} + 1)$. Now applying Prop. 3.1, we get a gap test for $\mathcal{B}$ by running this indicator $\tilde{O}(\frac{n}{b})$ times, so the cost of the gap test is $\tilde{O}(\sqrt{\frac{n \cdot \texttt{uloss}(\mathcal{B})}{b}} + \frac{n}{b})$. By hypothesis of Lemma 8.8, $b \geq \Omega(\texttt{uloss}(\mathcal{B}))$, so the running time can be bounded by $\tilde{O}(\sqrt{n} + \frac{n}{b})$.

In the case of $\mathbf{UGT}_1$, the argument is significantly more subtle. Again, we would like to construct a loss indicator for $\mathcal{B}$ obtained by running a loss indicator on a randomly chosen subbox of much narrower width. Unlike in $\mathbf{UGT}_2$, in this case we can't afford the running time of **PartialAlign**. For some width parameter r, consider boxes which are of the form $[i, i + r] \times [i - a, i + r + a]$. Given access to an $\texttt{Xloss}$-indicator for each of these boxes (which comes from $\mathbf{XLI}_1$), the procedure $\mathbf{XLI}_2$ constructs an $\texttt{Xloss}$-indicator by randomly sampling one of these boxes. (For technical reasons, we actually work on an enlarged box obtained from $\mathcal{B}$ by extending $\mathcal{B}$ r units in all directions and adding an increasing sequence of points of length r below and to the left and also above and to the right.) By choosing r to be a sufficiently large multiple of a, we ensure that this indicator can't be much smaller than $\frac{\texttt{Xloss}(\mathcal{B})}{w(\mathcal{B})}$. Unfortunately, when we use this indicator to get a gap test for $\texttt{Xloss}(\mathcal{B})$ using Prop. 3.1, the additive $\texttt{terr}_\varepsilon(\mathcal{B})$ term introduces an unacceptable additive error for $\texttt{Xloss}(\mathcal{B})$. Fortunately, when we do the straightforward conversion of the $\texttt{Xloss}(\mathcal{B})$ gap test into a $\texttt{uloss}(\mathcal{B})$ gap test, the additive error gets converted to an (acceptable) small multiplicative error in $\texttt{uloss}(\mathcal{B})$ (Again, the full details of this proof appear in section 9).

The running time analyses for $\mathbf{XLI}_2$ and $\mathbf{UGT}_1$ are trivial. The procedure $\mathbf{XLI}_2$ consists of one call to $\mathbf{XLI}_1$ on a box of width $O_{\varepsilon_3}(a)$. For input $\mathcal{B}, a, b$, the procedure $\mathbf{UGT}_1$ consists of $\widetilde{O}_{\varepsilon_3}(\frac{w(\mathcal{B})}{b})$ calls to $\mathbf{XLI}_2$. Using the bound on the running time of $\mathbf{XLI}_1$ established previously, we get the desired running time bound for $\mathbf{UGT}_1$.

6 Designing BGT

We pick up the discussion from section 3.2. We delve a little deeper into the ideas behind designing **BGT**. As mentioned earlier, we adapt the SS algorithm by implementing the specific kind of box queries. Unfortunately, the queries are approximate and we have to ensure that the SS guarantees are not affected (too much). We remind the reader of the definitions $d_{min}(\mathcal{B}) = \min(w(\mathcal{B}), h(\mathcal{B}))$ and $d_{max}(\mathcal{B}) = \max(w(\mathcal{B}), h(\mathcal{B}))$.

Given a box $\mathcal{B}$ and a budget t, we need to find t random points/matches inside this box, or certify that the box has at most $w(\mathcal{B})/t$ points in it. As mentioned earlier, we can choose $\widetilde{O}(t\sqrt{\mathcal{B}})$ uniform random samples from $x(\mathcal{B})$ and $y(\mathcal{B})$, count the matches among these, and scale up appropriately to estimate the actual number of matches. A direct birthday paradox argument shows that if $d_{min}(\mathcal{B}) = \Omega(d_{max}(\mathcal{B}))$, then this succeeds with high probability. Thus, when the box is "square", everything works with an additional running time factor of $\sqrt{\mathcal{B}}$.

We need to deal with the case when $d_{min}(\mathcal{B})$ is significantly smaller than $d_{max}(\mathcal{B})$, so the box is "thin". Since n is an upper bound on the dimensions of the box, we can choose to sample $\sqrt{n}$ samples from $x(\mathcal{B})$ and $y(\mathcal{B})$. By going through the birthday paradox calculation , we show that if $d_{min}(\mathcal{B}) = \Omega(\sqrt{n})$, then the procedure succeeds. If $d_{max}(\mathcal{B}) = O(\sqrt{n})$, then both dimensions of the box are small. We can exactly compute the number of matches in $O(\sqrt{n})$ time. (In terms of the input strings, we are provided two substrings of size $O(\sqrt{n})$ and wish to find the number of matches.)

This leaves us with the main problematic case: $d_{min}(\mathcal{B}) = o(\sqrt{n})$, $d_{max}(\mathcal{B}) = \omega(\sqrt{n})$. To deal with these boxes, we observe that the number of matches can be at most $d_{min}(\mathcal{B})$. If $d_{max}(\mathcal{B}) = w(\mathcal{B})$, then we can use this trivial upper bound. If $t \leq w(\mathcal{B})/h(\mathcal{B})$, we simply declare the box has few points in it. When $t \geq w(\mathcal{B})/h(\mathcal{B})$, we explicitly enumerate $y(\mathcal{B})$ and random sample $x(\mathcal{B})$ to discover matches.

Thus, the hard case is when $d_{min}(\mathcal{B}) = w(\mathcal{B})$. In an extreme case ($w(\mathcal{B}) = O(1)$, $h(\mathcal{B}) = \Omega(n)$), all x-indices in $\mathcal{B}$ could correspond to matches, yet random sampling cannot find a match. One may wonder why this is a problem, since the x and y axes of the points are completely symmetric. The problem is in the accounting of SS: losses can be measured with respect to $w(\mathcal{B})$ or $h(\mathcal{B})$, but it has to be done consistently for all boxes. Thus, if we commit to performing the SS analysis with respect to `Xloss`, we cannot "switch" the axes.

It requires some detailed understanding of SS to resolve this situation. We can consider square subboxes $\mathcal{B}'$ of $\mathcal{B}$ where $Y(\mathcal{B}')$ starts either at the beginning of $Y(\mathcal{B})$ or ends at the end of $Y(\mathcal{B})$. We can brute force within these subboxes. If we do not find any matches here, it turns out that we can afford to not find any matches in $\mathcal{B}$. In some sense, we can assert that $\mathcal{B}$ cannot have too many points on the final LCS, so erroneous answers for queries in $\mathcal{B}$ do not affect the final answer.

But all these approximations lead to a further technical difficulty. The final guarantee of **BGT** has an additive error as well, instead of a purely multiplicative error on the Ulam distance in $\mathcal{B}$. This additive error must now percolate through the entire speed-up process down to the final answer. This is why we define (τ, δ)-gap tests as opposed to (purely multiplicative) τ-gap tests. This requires generalizing all the speed-up analysis of AN to additive errors, introducing much extra notation and complications.

7 Road Map

Since this algorithm for ulam distance is very lengthy and technical with many moving parts, it may be helpful to the reader to have an outline of what pieces of the algorithm are covered in which sections and how they all fit together. In section 8, we state the "outer" procedures of the algorithm, consisting of the eight procedures (excluding $\mathbf{XLI}_0$) listed in section 3.3. These procedures are stated in the order in which they are called. Section 8 also contains statements of the lemmas we'll need for these procedures. These lemmas are proved in section 9.

From here, we move on to the discussion of the procedure $\mathbf{XLI}_0$ and the subprocedures it calls. Due to space limitations, we omit most of these details, instead providing a high level explanation of this process in section 10.

8 Algorithm

In this section, we state the eight procedures (excluding $\mathbf{XLI}_0$) referred to in section 3.3. Again, the procedure **PartialAlign** is restated from [5]. We first introduce the following definition:

Box Extension The a-extension of box $\mathcal{B}$, denoted by $\texttt{ext}_a(\mathcal{B})$ is the box obtained by concatenating two common subsequences of length a to the bottom left and top right corners of $\mathcal{B}$. Similarly, $\texttt{extL}_a(\mathcal{B})$ and $\texttt{extR}_a(\mathcal{B})$ are the boxes obtained by concatenating one common subsequence of length a to the bottom left corner and top right corner of $\mathcal{B}$ respectively. These boxes have the property $\texttt{uloss}(\mathcal{B}) = \texttt{uloss}(\texttt{extL}_a(\mathcal{B})) = \texttt{uloss}(\texttt{extR}_a(\mathcal{B})) = \texttt{uloss}(\texttt{ext}_a(\mathcal{B}))$.

We also introduce the primitive $\mathbf{Sample}(S, p)$. The goal of $\mathbf{Sample}(S, p)$ is to simulate choosing a subset

of a set S by selecting each element independently with probability p. It is known that this can be done in $\widetilde{O}(p|S|)$ time by choosing a value k subject to the binomial distribution with parameters $|S|$ and p and then selecting a random subset of S of size k. We will make use of this primitive in the procedure **BGT**.

For **PartialAlign**, $\beta = C_1 \log^3 n$ for some sufficiently large constant $C_1 > 0$, and $\gamma = C_2 \log n$ for some sufficiently large constant $C_2 > 0$.

The algorithm $\mathbf{UGT}_2$ calls the procedure **GTfromLI**, which is defined in section 3.1.

Main$(\mathcal{U}, \varepsilon)$
Output: Approximation to $\texttt{uloss}(\mathcal{U})$.
1. Fix global parameters (unchanged throughout algorithm).

Symbol	Value
ε_1	$\varepsilon/14$
ε_2	$\varepsilon_1/4$
ε_3	$\varepsilon_2/68$
ε_4	$\varepsilon_3/10$

2. For $i \leftarrow 0$ to $\log_{1+\varepsilon}(2n) - 1$
   (a) Run $\mathbf{UGT}_2(\mathcal{U}, \frac{2n}{(1+\varepsilon)^{i+1}}, \frac{2n}{(1+\varepsilon)^i})$ $O(\log n)$ times and take the majority answer.
   (b) If the majority answer is BIG, stop the i loop and return $\frac{2n}{(1+\varepsilon)^i}$.
3. If the i loop never stopped, return 0.

PartialAlign(A, B, d)
Output: Boxes $\mathcal{B}_0, \ldots, \mathcal{B}_k$
1. Split A and B into blocks of size βd. Set $m_0 = 0$, $k = \lceil \frac{n}{\beta d} \rceil$.
2. For $i \leftarrow 1$ to k
   (a) For $j \leftarrow 4$ to $\log 4n$
      (1) Pick a random location p in $[(i-1) \cdot \beta d + 4d, i \cdot \beta d - 4d]$.
      (2) Pick $\gamma \cdot 2^{j/2}$ random positions from each of $A[p, p + 2^j]$ and $B[p + m_{i-1} - 2^j, p + m_{i-1} + 2 \cdot 2^j]$.
      (3) If there is at least one collision $A[u] = B[v]$, then do the following.
         (i) Choose any such collision u_i, v_i. Set $m_i \leftarrow v_i - u_i, a_i \leftarrow u_i, b_i \leftarrow v_i$.
         (ii) Stop the j loop and jump to the next i.
   (b) If the j-loop did not stop, then fail.
3. Set $a_0 = b_0 = 0$, $a_{k+1} = |A|$, $b_{k+1} = |B|$.
4. Return the boxes $\mathcal{B}_0, \ldots, \mathcal{B}_k$, where $\mathcal{B}_i = (a_i, a_{i+1}] \times [b_i + 1, b_{i+1}]$.

$\mathbf{UGT}_2$ $(\mathcal{U}, a, b)$
Output: BIG or SMALL
1. $\mathcal{B}_0, \ldots, \mathcal{B}_k \longleftarrow$ **PartialAlign**$(X(\mathcal{U}), Y(\mathcal{U}), (1+\varepsilon)b)$.
2. Return the output of
GTfromLI$(a, b, \varepsilon_1, 1/15, \mathbf{ULI}_2(\mathcal{U}, a, b, \mathcal{B}_0, \ldots, \mathcal{B}_k))$.

$\mathbf{ULI}_2$ $(\mathcal{U}, a, b, \mathcal{B}_0, \ldots, \mathcal{B}_k)$
Output: 0 or 1
1. Choose $i \in [k] \cup \{0\}$ at random with probability $\frac{w(\mathcal{B}_i)+h(\mathcal{B}_i)}{w(\mathcal{U})+h(\mathcal{U})}$.
2. Return $\mathbf{ULI}_1(\mathcal{B}_i)$.

$\mathbf{ULI}_1$ $(\mathcal{B})$
Output: 0 or 1
1. Set $a = \max(|w(\mathcal{B}) - h(\mathcal{B})|, 1), b = w(\mathcal{B}) + h(\mathcal{B})$.
2. while $\frac{b}{a} > 1 + \varepsilon_2$ do
   (a) Run $\mathbf{UGT}_1(\mathcal{B}, a, b)$ $O(\log n)$ times.
   (b) If the majority answer is BIG, set $a = a(1+\varepsilon_2)$.
   (c) Else with probability $\frac{\varepsilon_2}{1+\varepsilon_2}$ return 0 and halt.
   (d) Else set $b = \frac{b}{1+\varepsilon_2}$ and continue.
3. If $a > 1$ or $w(\mathcal{B}) \neq h(\mathcal{B})$, return 1.
4. Else return 0.

$\mathbf{UGT}_1$ $(\mathcal{B}, a, b)$
Output: BIG or SMALL
1. If $b < (1 + \varepsilon_3)(h(B) - w(B))$ return BIG.
2. Set $a' = \lceil a \rceil$, $r = \lceil \frac{100}{\varepsilon_3^2} \rceil a'$.
3. Run $\mathbf{XLI}_2(\mathcal{B}, a', r)$ $k = \frac{24 w(\mathcal{B})}{b(\varepsilon_3)^3}$ times and let z be the sum of the outputs.
4. If $z > (1 - 6\varepsilon_3)\frac{k}{w(\texttt{extL}_r(\mathcal{B}))}\frac{b+w(\mathcal{B})-h(\mathcal{B})}{2}$, return BIG. Otherwise, return SMALL.

$\mathbf{XLI}_2$ $(\mathcal{B}, a', r)$
Output: 0 or 1
1. Set $x = 0$.
2. Let p_1 be a uniformly random element of $X(\texttt{extL}_r(\mathcal{B}))$, and let $p_2 = p_1 + r$.
3. Let $\mathcal{T}$ be the subbox of $\texttt{ext}_r(\mathcal{B})$ given by $(p_1, p_2] \times [p_1 - a', p_2 + a']$.
4. Return $\mathbf{XLI}_1(\mathcal{T})$.

$\mathbf{XLI}_1$ $(\mathcal{T})$
Output: 0 or 1
1. Set $i = 0$
2. While $i \leq \log_{1+\varepsilon_3}(w(\mathcal{T})) + 1$ do
   (a) Run $\mathbf{BGT}(\mathcal{T}, \frac{w(\mathcal{T})}{(1+\varepsilon_3)^{i+1}}, \frac{w(\mathcal{T})}{(1+\varepsilon_3)^i})$ $O(\log n)$ times and take the majority.
   (b) If the majority answer is BIG, return 1 and halt.
   (c) Else with probability $\frac{\varepsilon_3}{1+\varepsilon_3}$ return 0 and halt. (Otherwise continue)
   (d) Set $i = i + 1$.
3. Return 0.

BGT$(\mathcal{T}, a, b)$
Output: BIG or SMALL
1. Approximate the number of X-indices of $\mathcal{T}$ whose match lies outside $\mathcal{T}$ as follows:
(a) If $b \leq \frac{16(2e-1)}{\varepsilon_4^2}\sqrt{w(\mathcal{T})}\log n$
• Let V be the number of characters in $f_X(X(\mathcal{T}))$ not contained in $f_Y(Y(\mathcal{T}))$, obtained by reading the intervals entirely.
(b) If $b > \frac{16(2e-1)}{\varepsilon_4^2}\sqrt{w(\mathcal{T})}\log n$
• Set $p = \min(O(\frac{\sqrt{w(\mathcal{T})}\log n}{b}), 1)$.
• Set $\hat{X} =$ **Sample**$(X(\mathcal{T}), p)$, $\hat{Y} =$ **Sample**$(Y(\mathcal{T}), p)$
• Let M be the number of collisions between $\hat{X}$ and $\hat{Y}$.
• Let $V = w(\mathcal{T}) - \frac{M}{p^2}$.
2. Set $r = \min(O(\frac{\log n}{\sqrt{b}}), 1)$. Run **Sample**$(X(\mathcal{T}), r)$ and **Sample**$(Y(\mathcal{T}), r)$ and find the collisions.
3. Run $\mathbf{XLI}_0(x, \mathcal{T})$ $O(\log n)$ times on each point x and take the majority outcome. Let D be the number of points classified as bad, and let $W = \frac{D}{r^2}$.
4. If $V+W \geq b(1-\varepsilon_4)$, return BIG, else return SMALL.

We state the nine main lemmas, which refer to the nine algorithms listed above (as well as $\mathbf{XLI}_0$). These lemmas correspond to the nine steps listed in section 3.3.

For convenience, we will state the guarantees that we will prove in Tab. 1. Tab. 1 lists each of the 9 procedures along with the running time, quality, failure probability and any constraints on the input that are assumed in establishing the behavior. For each procedure there is a lemma that establishes the guarantees for that procedure, assuming the guarantees for the procedure listed immediately before it in the table.

By adapting the SS algorithm, we obtain the following procedure $\mathbf{XLI}_0$. $\mathbf{XLI}_0$ has a multiplicative quality term ε_4, as well as an additive term $\texttt{terr}_{\varepsilon_3}(\mathcal{T})$, which (as mentioned earlier) is a somewhat technical term whose definition we defer to section 9.

LEMMA 8.1. *Suppose $\mathcal{T}$ satisfies the input constraints for $\mathbf{XLI}_0(x, \mathcal{T})$ in Tab. 1. Then for x chosen uniformly at random from $X(\mathcal{T})$, $\mathbf{XLI}_0(x, \mathcal{T})$ is an* `Xloss`*-indicator for $\mathcal{T}$ achieving the guarantees listed in Tab. 1.*

LEMMA 8.2. *Assume that for any $\mathcal{T}$ satisfying the input constraints for $\mathbf{XLI}_0$, running $\mathbf{XLI}_0(x, \mathcal{T})$ for x chosen uniformly at random from $X(\mathcal{T})$ is an* `Xloss`*-indicator for $\mathcal{T}$ achieving the guarantees listed in Tab. 1. Then $\mathbf{BGT}(\mathcal{T}, a, b)$ is a gap test for* `Xloss`$(\mathcal{T})$ *achieving the guarantees listed in Tab. 1.*

LEMMA 8.3. *Assume that for any box $\mathcal{T}$ satisfying the input constraints of* **BGT** *in Tab. 1, $\mathbf{BGT}(\mathcal{T}, a, b)$ is a gap test for* `Xloss`$(\mathcal{T})$ *achieving the guarantees listed in Tab. 1. Then $\mathbf{XLI}_1(\mathcal{T})$ is an* `Xloss`*-indicator for $\mathcal{T}$ achieving the guarantees listed in Tab. 1.*

LEMMA 8.4. *Suppose that $\mathcal{B}, a', r$ satisfy the input constraints for $\mathbf{XLI}_2(\mathcal{B}, a', r)$ in Tab. 1. Assume that for any box $\mathcal{T}$ satisfying the input constraints of $\mathbf{XLI}_1$ in Tab. 1, $\mathbf{XLI}_1(\mathcal{T})$ is an* `Xloss`*-indicator achieving the guarantees in Tab. 1. Then $\mathbf{XLI}_2(\mathcal{B}, a', r)$ is an* `Xloss`*-indicator for $\mathcal{T}$ achieving the guarantees listed in Tab. 1.*

One should note here that the quality guarantee on $\boldsymbol{XLI}_2$ depends on whether the input parameter a' is at least `uloss`$(\mathcal{B})$. If $a' \geq$ `uloss`$(\mathcal{B})$ then the probability that $\boldsymbol{XLI}_2$ is 1 is tightly bounded around `Xloss`$(\mathcal{B})$. However if $a' <$ `uloss`$(\mathcal{B})$ then we only have a *one-sided guarantee*: we are assured that the probability that $\boldsymbol{XLI}_2$ is 1 is not much smaller than `Xloss`$(\mathcal{B})$ but we have no upper bound guarantee on this probability (for example, the output might be 1 with probability 1).

LEMMA 8.5. *Assume that for $\mathcal{B}, a', r$ satisfying the input constraints for $\mathbf{XLI}_2$ in Tab. 1, $\mathbf{XLI}_2(\mathcal{B}, a', r)$ is an* `Xloss`*-indicator (with upper bound $w(extL_r(\mathcal{B}))$) achieving the guarantees in Tab. 1. Then $\mathbf{UGT}_1(\mathcal{B}, a, b)$ is a gap test for* `uloss`$(\mathcal{B})$ *achieving the guarantees in Tab. 1.*

LEMMA 8.6. *Suppose that for a box $\mathcal{B}$, $\mathbf{UGT}_1(\mathcal{B}, a, b)$ is a gap test for* `uloss`$(\mathcal{B})$ *achieving the guarantees Tab. 1. Then $\mathbf{ULI}_1(\mathcal{B})$ is a* `uloss`*-indicator achieving the guarantees Tab. 1.*

LEMMA 8.7. *Suppose that $\mathcal{U}, a, b, \mathcal{B}_0, \ldots, \mathcal{B}_k$ satisfy the input constraints for $\mathbf{ULI}_2$ in Tab. 1. Assume that for any box $\mathcal{B}$, $\mathbf{ULI}_1(\mathcal{B})$ is a* `uloss`*-indicator achieving the guarantees in Tab. 1. Then $\mathbf{ULI}_2(\mathcal{U}, a, b, \mathcal{B}_0, \ldots, \mathcal{B}_k)$ is a* `uloss`*-indicator achieving the guarantees in Tab. 1.*

LEMMA 8.8. *Suppose that $\mathcal{U}, a, b$ satisfy the input constraints for* UGap *in Tab. 1. Assume that, for $\mathcal{U}, a, b, \mathcal{B}_0, \ldots, \mathcal{B}_k$ satisfying the input constraints for $\mathbf{ULI}_2$ in Tab. 1, $\mathbf{ULI}_2(\mathcal{U}, a, b, \mathcal{B}_0, \ldots, \mathcal{B}_k)$ is a* `uloss`*-indicator achieving the guarantees in Tab. 1. Then $\mathbf{UGT}_2(\mathcal{U}, a, b)$ is a gap test for* `uloss`$(\mathcal{U})$ *achieving the guarantees in Tab. 1.*

LEMMA 8.9. *Suppose that for $\mathcal{U}, a, b$ satisfying the input constraints of $\mathbf{UGT}_2$ in Tab. 1, $\mathbf{UGT}_2(\mathcal{U}, a, b)$ is a gap test for* `uloss`$(\mathcal{U})$ *achieving the guarantees in Tab. 1. Then with probability at least 2/3,* **Main**$(\mathcal{U}, \varepsilon)$ *outputs an $(\varepsilon, 0)$ approximation to* `uloss`$(\mathcal{U})$ *and runs in time $\widetilde{O}_\varepsilon(\frac{n}{\texttt{uloss}(\mathcal{U})} + \sqrt{n})$.*

Table 1: Procedure Guarantees

	Procedures				
	$\mathbf{XLI}_0(x, \mathcal{T})$	$\mathbf{BGT}(\mathcal{T}, a, b)$	$\mathbf{XLI}_1(\mathcal{T})$	$\mathbf{XLI}_2(\mathcal{B}, a', r)$	$\mathbf{UGT}_1(\mathcal{B}, a, b)$
Running Time	$\widetilde{O}_{\varepsilon_3}(\sqrt{w(\mathcal{T})})$	$\widetilde{O}_{\varepsilon_3}(\frac{w(\mathcal{T})^{3/2}}{b})$	$\widetilde{O}_{\varepsilon_3}(\sqrt{w(\mathcal{T})})$	$\widetilde{O}_{\varepsilon_3}(\sqrt{r})$	$\widetilde{O}_{\varepsilon_2}(\frac{w(\mathcal{B})\sqrt{a}}{b})$
Quality	$(0, \varepsilon_4 \frac{\texttt{Xloss}(\mathcal{T})}{w(\mathcal{T})} + (1+\varepsilon_4)\frac{\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})})$	$(\varepsilon_3, \texttt{terr}_{\varepsilon_3}(\mathcal{T}))$	$(\frac{2\varepsilon_3}{1+2\varepsilon_3}\frac{\texttt{Xloss}(\mathcal{T})}{w(\mathcal{T})}, \varepsilon_3\frac{\texttt{Xloss}(\mathcal{T})+1/n}{w(\mathcal{T})} + (1+\varepsilon_3)\frac{\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})})$	$(\frac{5\varepsilon_3\texttt{Xloss}(\mathcal{B})}{w(extL_r(\mathcal{B}))}, \nu)$ where $\nu = 5\varepsilon_3\frac{\texttt{Xloss}(\mathcal{B})+1}{w(extL_r(\mathcal{B}))} + 2\varepsilon_3\frac{h(\mathcal{B})-w(\mathcal{B})}{w(extL_r(\mathcal{B}))}$ if $a' \geq \texttt{uloss}(\mathcal{B})$ and $\nu = 1$ otherwise.	ε_2
Failure Probability	1/3	1/3	0	0	1/3
Input Constraints	$h(\mathcal{T}) \leq 2w(\mathcal{T})$	$h(\mathcal{T}) \leq 2w(\mathcal{T})$, $b \geq (1+\varepsilon_3)a$	$h(\mathcal{T}) \leq 2w(\mathcal{T})$	$r \geq \lceil \frac{100a'}{\varepsilon_3^2} \rceil$	$b \geq (1+\varepsilon_2)a$, $a \geq 1$
	Procedures				
	$\mathbf{ULI}_1(\mathcal{B})$	$\mathbf{ULI}_2(\mathcal{U}, a, b, \mathcal{B}_0, \ldots, \mathcal{B}_k)$	$\mathbf{UGT}_2(\mathcal{U}, a, b)$	$\mathbf{Main}(\mathcal{U}, \varepsilon)$	
Running Time	$\widetilde{O}_{\varepsilon_1}(\sqrt{\texttt{uloss}(\mathcal{B})}+1)$	$\widetilde{O}_{\varepsilon_1}(\sqrt{\frac{b\texttt{uloss}(\mathcal{U})}{n}}+1)$	$\widetilde{O}_{\varepsilon}(\frac{n}{b}+\sqrt{n})$	$\widetilde{O}_{\varepsilon}(\frac{n}{\texttt{uloss}(\mathcal{U})}+\sqrt{n})$	
Quality	$(\frac{\varepsilon_1}{1+\varepsilon_1}\frac{\texttt{uloss}(\mathcal{B})}{w(\mathcal{B})+h(\mathcal{B})}, \varepsilon_1\frac{\texttt{uloss}(\mathcal{B})+1/n}{w(\mathcal{B})+h(\mathcal{B})})$	$(\frac{\varepsilon_1}{1+\varepsilon_1}\frac{\texttt{uloss}(\mathcal{U})}{w(\mathcal{U})+h(\mathcal{U})}, \varepsilon_1\frac{\texttt{uloss}(\mathcal{U})+1}{w(\mathcal{U})+h(\mathcal{U})})$	ε	ε	
Failure Probability	0	0	2/5	1/3	
Input Constraints		$\mathcal{U}$ is an $n \times n$ box, $\mathcal{B}_0, \ldots, \mathcal{B}_k$ is a box chain spanning $\mathcal{U}$ with $\sum_{i=0}^{k} \texttt{uloss}(\mathcal{B}_i) = \texttt{uloss}(\mathcal{U})$ and $\forall i \in [k]\cup\{0\}, w(\mathcal{B}_i), h(\mathcal{B}_i) = \widetilde{O}(b)$	$\mathcal{U}$ is an $n \times n$ box with $\texttt{uloss}(\mathcal{U}) \leq (1+\varepsilon)b$, $b \geq (1+\varepsilon)a$	$\mathcal{U}$ is an $n \times n$ box	

9 Analysis

In this section, we provide proofs of eight of the nine lemmas stated in section 3.3 (Due to space limitations, we omit the full proof of Lemma 8.1). We also state and prove several other lemmas, which are tools that help us prove these eight lemmas. In order to do so, we first formally define the function `terr`, as well as several other bits of necessary notation.

9.1 The function terr

In section 2.1, we introduced the loss functions `uloss` and `Xloss`. For the analysis, we will need to extend the definition of `Xloss` to sequences, and define another loss function, $\texttt{Yloss}_{\texttt{trim}}$.

As stated in section 2.1, $\texttt{Xloss}(\mathcal{B}) = w(\mathcal{B}) - \texttt{lcs}(\mathcal{B})$, which is the number of X-indices of $\mathcal{B}$ that are not matched by the LCS. We extend this function to a sequence S, where $\texttt{Xloss}(S, \mathcal{B}) = w(\mathcal{B}) - |S|$. We will often omit the $\mathcal{B}$ input when it is clear.

The function $\texttt{Yloss}_{\texttt{trim}}$. Let S be an increasing point sequence in a box $\mathcal{B}$ with $|S| = k$. For P_1, P_k the first and last points of S respectively, $\texttt{Yloss}_{\texttt{trim}}(S, \mathcal{B}) = (y(P_k) - y(P_1)) - (k-1)$. Intuitively, $\texttt{Yloss}_{\texttt{trim}}(S, \mathcal{B})$ represents the minimum number of Y-indices that can be missed by any increasing sequence formed by concatenating S with other increasing sequences on the left or right. Note that if S is an LCS of $\mathcal{B}$, then

$\texttt{Xloss}(\mathcal{B}) + \texttt{Yloss}_{\texttt{trim}}(S, \mathcal{B}) \leq \texttt{uloss}(\mathcal{B})$. Again, we will often omit the $\mathcal{B}$ input when it is clear.

With these tools, we now provide the notion of an *error function*. An error function for a sequence S and box $\mathcal{B}$ outputs a nonnegative real number. Error functions are used to measure differences between loss functions of different common subsequences in $\mathcal{B}$. Intuitively, we use error functions to compare the length of the sequences our algorithm finds with the length of the LCS. This is how we keep track of and control the quality of our algorithm.

We will use the following error functions:

- **The function herr.** $\texttt{herr}(S, \mathcal{B}) = \texttt{Xloss}(S, \mathcal{B}) - \texttt{Xloss}(\mathcal{B}) = \texttt{lcs}(\mathcal{B}) - |S|$.
- **The function serr.** $\texttt{serr}_\varepsilon(S, \mathcal{B}) = \min_{S' \subseteq S}(\texttt{herr}(S', \mathcal{B}) + \varepsilon \cdot \texttt{Yloss}_{\texttt{trim}}(S', \mathcal{B}))$.
- **The function terr.** $\texttt{terr}_\varepsilon(\mathcal{B}) = \max_S(\texttt{serr}_\varepsilon(S, \mathcal{B}))$, where the max is taken over common subsequences S of $\mathcal{B}$ with length at least $\texttt{lcs}(\mathcal{B}) - \varepsilon \cdot \texttt{Xloss}(\mathcal{B})$.

9.2 Proofs We begin by proving Lemma 8.9.

Proof. The procedure **Main** is a loop that performs $O(\frac{1}{\varepsilon}\log n)$ iterations. In each iteration, $\mathbf{UGT}_2$ is called $O(\log n)$ times and we take a majority vote on the answers. Each iteration is said to be *reliable* if the majority vote is correct, and the run of **Main** is *reliable* provided that each of the iterations is reliable. We will show that if the run is reliable then the estimate provided by **Main** has the desired properties. First we observe that the probability that the run is unreliable is small. By the Chernoff Bound, since $\mathbf{UGT}_2$ is correct with probability at least 3/5. the probability that the majority of $O(\log n)$ runs of $\mathbf{UGT}_2$ is wrong is bounded by $\frac{1}{n^6}$. By the union bound over all iterations, the probability that the run is unreliable is at most $\frac{1}{\varepsilon n^5}$.

We now assume that the run is reliable. The algorithm successively takes the majority of $O(\log n)$ independent runs of $\mathbf{UGT}_2$ with lower threshold $\frac{2n}{(1+\varepsilon)^{i+1}}$ and upper threshold $\frac{2n}{(1+\varepsilon)^i}$ for increasing values of i and stops and outputs the estimate $\frac{2n}{(1+\varepsilon)^i}$ the first time the majority vote of the $\mathbf{UGT}_2$ tests returns BIG. If no iteration returns BIG, the output is 0. We consider two cases: $\texttt{uloss}(\mathcal{U}) = 0$ and $\texttt{uloss}(\mathcal{U}) \geq 1$.

If $\texttt{uloss}(\mathcal{U}) = 0$, then every test returns SMALL, and so the output will be 0.

Otherwise $\texttt{uloss}(\mathcal{U}) \geq 1$. Let k be the index such that $\frac{2n}{(1+\varepsilon)^{k+1}} < \texttt{uloss}(\mathcal{U}) \leq \frac{2n}{(1+\varepsilon)^k}$. During iteration j, a reliable run will return SMALL for $j \leq k-1$ and BIG for $j \geq k+1$. It might return either SMALL or BIG for $j = k$. Therefore the output will be either $\frac{2n}{(1+\varepsilon)^{k+1}}$ or $\frac{2n}{(1+\varepsilon)^k}$. By the choice of k this is at least $\frac{\texttt{uloss}(\mathcal{U})}{1+\varepsilon}$ and at most $(1+\varepsilon)\texttt{uloss}(\mathcal{U})$, which is an $(\varepsilon, 0)$ approximation to $\texttt{uloss}(\mathcal{U})$.

To analyze the running time, we simply see that each iteration of the loop involves running $\mathbf{UGT}_2$ $O(\log n)$ times, which is in $\widetilde{O}_\varepsilon(\frac{n}{n/(1+\varepsilon)^i} + \sqrt{n})$ time for each i. Since the loop stops when $\frac{2n}{(1+\varepsilon)^i} = \widetilde{O}_\varepsilon(\texttt{uloss}(\mathcal{U}))$, the loop runs in time $\widetilde{O}_\varepsilon(\frac{n}{\texttt{uloss}(\mathcal{U})} + \sqrt{n})$, as required. □

We restate Lemma 3.1 from [5], which establishes the necessary properties of the procedure **PartialAlign**.

LEMMA 9.1. *Suppose $\mathcal{U}$ is an $n \times n$ box with $\texttt{uloss}(\mathcal{U}) \leq d$. Let $\mathcal{B}_0, \ldots, \mathcal{B}_k$ be the output of* **PartialAlign**$(X(\mathcal{U}), Y(\mathcal{U}), d)$. *Then with probability at least 2/3, the following all hold.*

1. *The boxes $\mathcal{B}_0, \ldots, \mathcal{B}_k$ form a box chain spanning $\mathcal{U}$ with $\sum_{i=0}^{k} \texttt{uloss}(\mathcal{B}_i) = \texttt{uloss}(\mathcal{U})$.*
2. *For any i, $w(\mathcal{B}_i), h(\mathcal{B}_i) = O(d \log^3 n)$.*
3. *$k = O(\frac{n}{d \log^3 n})$*
4. *The running time of* **PartialAlign**$(X(\mathcal{U}), Y(\mathcal{U}), d)$ *is $\tilde{O}(\frac{n}{d} + \sqrt{n})$.*

(We mention that the upper bound on k in our restatement is not explicitly in Lemma 3.1 of [5], but is implicit in the definition of the procedure.)

Intuitively, what **PartialAlign** accomplishes (with high probability) is to build a box chain inside $\mathcal{U}$ with the property that the LCS of $\mathcal{U}$ is just the concatenation of the LCS's of each of the boxes in the box chain.

Using Lemma 9.1, we prove Lemma 8.8.

Proof. First, we note that Lemma 9.1 ensures that with probability at least 2/3, $\mathcal{B}_0, \ldots, \mathcal{B}_k$ satisfy the input constraints for $\mathbf{ULI}_2$ in Tab. 1. Therefore by assumption, $\mathbf{ULI}_2(\mathcal{U}, a, b, \mathcal{B}_0, \ldots, \mathcal{B}_k)$ is a $(\frac{\varepsilon_1}{1+\varepsilon_1}\frac{\texttt{uloss}(\mathcal{U})}{w(\mathcal{U})+h(\mathcal{U})}, \varepsilon_1\frac{\texttt{uloss}(\mathcal{U})+1}{w(\mathcal{U})+h(\mathcal{U})})$-quality $\texttt{uloss}$-indicator for $\mathcal{U}$. Therefore, by Prop. 3.1, $\mathbf{GTfromLI}(a, b, \varepsilon_1, 1/15, \mathbf{ULI}_2(\mathcal{U}, a, b, \mathcal{B}_0, \ldots, \mathcal{B}_k))$ is a $(\varepsilon_1, 2\varepsilon_1(\texttt{uloss}(\mathcal{U}) + 1))$-gap test, which by Prop. 3.2 is a $5\varepsilon_1$-gap test for $\texttt{uloss}(\mathcal{U})$. Furthermore, the failure probability of this gap test at most 1/15. Combining this via a union bound with the probability that **PartialAlign** fails, $\mathbf{UGT}_2$ succeeds with probability at least 3/5.

To analyze the running time, we see that $\mathbf{UGT}_2(\mathcal{U}, a, b)$ consists of first running **PartialAlign** and making $\widetilde{O}_{\varepsilon_1}(\frac{n}{b})$ calls to $\mathbf{ULI}_2$. **PartialAlign** runs

in time $\widetilde{O}(\frac{n}{b}+\sqrt{n})$, and by assumption, $\mathbf{ULI}_2$ runs in time $\widetilde{O}_{\varepsilon_1}(\sqrt{\frac{b\mathtt{uloss}(\mathcal{U})}{n}}+1)$. Since $b=\Omega(\mathtt{uloss}(\mathcal{U}))$ by assumption, this is at most $\widetilde{O}_{\varepsilon_1}(\frac{b}{\sqrt{n}}+1)$. Therefore, the total running time of all $\widetilde{O}_{\varepsilon_1}(\frac{n}{b})$ calls to $\mathbf{ULI}_2$ is $\widetilde{O}_{\varepsilon_1}(\frac{n}{b}+\sqrt{n})$. Combining this with the running time of **PartialAlign** yields the running time guarantee for $\mathbf{UGT}_2$ stated in Tab. 1.

We now prove Lemma 8.7.

Proof. We want $\mathbf{ULI}_2$ to be a good $\mathtt{uloss}$-indicator. It is obtained by using the box chain given by **PartialAlign**, and selecting a box at random (proportional to its width plus height) and using the $\mathtt{uloss}$-indicator $\mathbf{ULI}_1$ for that box.

We first prove that $\mathbf{ULI}_2$ is a good enough $\mathtt{uloss}$-indicator. For ease of notation, write $\Delta(\mathcal{B})=w(\mathcal{B})+h(\mathcal{B})$. Assume that $\mathcal{U},a,b,\mathcal{B}_0,\ldots,\mathcal{B}_k$ satisfy the input constraints for $\mathbf{ULI}_2$ in Tab. 1. By the assumption on $\mathbf{ULI}_1$, for any $\mathcal{B}_i$, the probability that $\mathbf{ULI}_1(\mathcal{B}_i)$ outputs 1 is between $\frac{1}{1+\varepsilon_1}\frac{\mathtt{uloss}(\mathcal{B}_i)}{\Delta(\mathcal{B})}$ and $(1+\varepsilon_1)\frac{\mathtt{uloss}(\mathcal{B}_i)}{\Delta(\mathcal{B}_i)}+\frac{\varepsilon_1}{n\Delta(\mathcal{B}_i)}$. Since $\mathcal{B}_i$ is chosen with probability $\frac{\Delta(\mathcal{B}_i)}{\Delta(\mathcal{U})}$, by taking a weighted average of the upper and lower bounds for the probability that $\mathbf{ULI}_1(\mathcal{B}_i)=1$ we get the that probability that $\mathbf{ULI}_2(\mathcal{U})$ is 1 is between $\frac{1}{1+\varepsilon_1}\frac{\mathtt{uloss}(\mathcal{U})}{\Delta(\mathcal{U})}$ and $(1+\varepsilon_1)\frac{\mathtt{uloss}(\mathcal{U})}{\Delta(\mathcal{U})}+\frac{\varepsilon_1}{\Delta(\mathcal{U})}$, as required.

Before bounding the expected running time, we point out that when we use **PartialAlign** the parameter d in **PartialAlign** is $\Theta(b)$ and so $\Delta(\mathcal{B}_i)=\widetilde{O}(b)$ and $k=\widetilde{O}(n/b)$. The expected running time of $\mathbf{ULI}_2$ is the average of the running time of $\mathbf{ULI}_1(\mathcal{B}_i)$ on the chosen input $\mathcal{B}_i$. By assumption, $\mathbf{ULI}_1(\mathcal{B}_i)$ has expected running time $\widetilde{O}_{\varepsilon_1}(\sqrt{\mathtt{uloss}(\mathcal{B}_i)}+1)$. Averaging over the selection of $\mathcal{B}_i$, the expected running time of $\mathbf{ULI}_2$ is $\sum_{i=0}^{k}\frac{\Delta(\mathcal{B}_i)}{\Delta(\mathcal{U})}\widetilde{O}_{\varepsilon_1}(\sqrt{\mathtt{uloss}(\mathcal{B}_i)}+1)$, which is at most $\frac{\widetilde{O}(b)}{\Delta(\mathcal{U})}\sum_{i=0}^{k}\widetilde{O}_{\varepsilon_1}(\sqrt{\mathtt{uloss}(\mathcal{B}_i)}+1)$, since $\Delta(\mathcal{B}_i)=\widetilde{O}(b)$. Using the concavity of the square root function and noting that the average value of $\mathtt{uloss}(\mathcal{B}_i)$ is $\mathtt{uloss}(\mathcal{U})/k$, $\widetilde{O}(\sum_{i=0}^{k}(\sqrt{\mathtt{uloss}(\mathcal{B}_i)}+1))\le\widetilde{O}(\sqrt{k\mathtt{uloss}(\mathcal{U})}+k))$. Since $k=\widetilde{O}(\frac{n}{b})$, we get

$$\frac{\widetilde{b}}{\Delta(\mathcal{U})}\sum_{i=0}^{k}\widetilde{O}_{\varepsilon_1}(\sqrt{\mathtt{uloss}(\mathcal{B}_i)}+1)$$
$$\le\widetilde{O}_{\varepsilon_1}(\frac{b}{n}(\sqrt{k\mathtt{uloss}(\mathcal{U})}+k))\le\widetilde{O}_{\varepsilon_1}(\sqrt{\frac{b\mathtt{uloss}(\mathcal{U})}{n}}+1).$$

This establishes the running time guarantee of $\mathbf{ULI}_2$ in Tab. 1. □

We now prove Lemma 8.6. We need to show that $\mathbf{ULI}_1(\mathcal{B})$ gives a sufficiently good $\mathtt{uloss}$-indicator provided that $\mathbf{UGT}_1$ is a sufficiently good gap test.

Proof. $\mathbf{ULI}_1(\mathcal{B})$ seeks to output 1 with probability approximately equal to $\mathtt{uloss}(\mathcal{B})/(w(\mathcal{B})+h(\mathcal{B}))$ and output 0 otherwise. The procedure involves a sequence of iterations. In each iteration, the procedure takes a majority vote of $O(\log n)$ instances of the gap test $\mathbf{UGT}_1$ with a particular lower threshold a and upper threshold b. If the iteration returns BIG, the value of a is increased by a factor of $(1+\varepsilon)$ while if the iteration returns SMALL, the value of b is decreased by the same factor. These new values of a and b are used in the next iteration. In the case that the iteration returns SMALL, the algorithm may stop and return 0 with a small probability. The process continues until a and b are close enough, in which case the output is 1.

We say that each iteration is reliable if the majority of $O(\log n)$ runs of $\mathbf{UGT}_1$ is correct, and $\mathbf{ULI}_1(\mathcal{B})$ is reliable if all of the iterations are reliable. By a Chernoff bound, each iteration is unreliable with probability at most $\frac{1}{n^6}$. By a union bound, using the fact that the number of iterations is at most $O(\frac{1}{\varepsilon_2}\log n)$ the probability that the run of $\mathbf{ULI}_1$ is unreliable is at most $\frac{1}{\varepsilon_2 n^5}$. For $\varepsilon_2\ge\frac{1}{n^2}$, this is at most $\frac{1}{n^3}$.

The correctness condition for $\mathbf{ULI}_1$is that the probability that 1 is output is within certain bounds. We now compute the probability that the output is 1 conditioned on the run being reliable. We then have to make a slight adjustment because of the chance it is unreliable.

So assume that the run of $\mathbf{ULI}_1$ is reliable. First suppose that $\mathtt{uloss}(\mathcal{B})>1$. Let j be such that $\frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^{j+1}}<\mathtt{uloss}(\mathcal{T})\le\frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^{j}}$. If $b\le\frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^{j+1}}$, then the test on line 2a will return BIG. If $a\ge\frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^{j}}$, then the test on line 2a will return SMALL. Each time the test on line 2a returns SMALL, b decreases by a factor of $(1+\varepsilon_2)$. If this happens $j+1$ times and each time the procedure does not return 0, the test on line 2a will continue to return BIG until $\frac{b}{a}\le1+\varepsilon_2$, at which point it will return 1 from line 3. If at some point $a\ge\frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^{j}}$, then the test on line 2a will continue to return SMALL until it returns 0 or b reaches $\frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^{j-1}}$, at which point the loop will stop and the procedure will return 1 from line 3. Therefore, if $\mathtt{uloss}(\mathcal{B})>1$, $\mathbf{ULI}_1(\mathcal{B})$ will return 1 iff line 2c never returns 0, which happens with probability $\frac{\varepsilon_2}{1+\varepsilon_2}$ independently each of k times, where k is between $j-1$ and $j+1$. Therefore, the probability that it returns 1 is between $\frac{\mathtt{uloss}(\mathcal{B})}{(1+\varepsilon_2)(w(\mathcal{B})+h(\mathcal{B}))}$ and $\frac{(1+\varepsilon_2)^2\mathtt{uloss}(\mathcal{B})}{w(\mathcal{B})+h(\mathcal{B})}$.

If $\mathtt{uloss}(\mathcal{B})=1$, then if line 2c never returns 0, when the loops ends, a will be either 1 or $1+\varepsilon_2$. In either

case, the procedure will return 1 via either line 3 or line 4. This will happen with probability between $\frac{1}{w(\mathcal{B})+h(\mathcal{B})}$ and $\frac{(1+\varepsilon_2)^2}{w(\mathcal{B})+h(\mathcal{B})}$, which is again between $\frac{\texttt{uloss}(\mathcal{B})}{(1+\varepsilon_2)(w(\mathcal{B})+h(\mathcal{B}))}$ and $\frac{(1+\varepsilon_2)^2\texttt{uloss}(\mathcal{B})}{w(\mathcal{B})+h(\mathcal{B})}$.

Finally, if $\texttt{uloss}(\mathcal{B}) = 0$, then line 2a will never return BIG, and the procedure will return 0, either on line 2c or on line 4, meaning the probability it returns 1 is $\frac{\texttt{uloss}(\mathcal{B})}{w(\mathcal{B})+h(\mathcal{B})}$.

Combining these bounds via a union bound with the chance that the run is unreliable, the probability that $\mathbf{ULI}_1(\mathcal{B})$ returns 1 is between $\frac{\texttt{uloss}(\mathcal{B})}{(1+\varepsilon_2)(w(\mathcal{B})+h(\mathcal{B}))} - \frac{1}{n^3}$ and $\frac{(1+\varepsilon_2)^2\texttt{uloss}(\mathcal{B})}{w(\mathcal{B})+h(\mathcal{B})} + \frac{1}{n^3}$. If $\texttt{uloss}(\mathcal{B}) = 0$, then since the probability of returning 1 is at least 0, we can replace the lower bound with $\frac{\texttt{uloss}(\mathcal{B})}{(1+2\varepsilon_2)(w(\mathcal{B})+h(\mathcal{B}))}$. If $\texttt{uloss}(\mathcal{B}) \geq 1$, then since $\frac{1}{n^3} \geq \frac{\varepsilon_2}{(w(\mathcal{B})+h(\mathcal{B}))(1+\varepsilon_2)(1+2\varepsilon_2)}$, we can again replace the lower bound with $\frac{\texttt{uloss}(\mathcal{B})}{(1+2\varepsilon_2)(w(\mathcal{B})+h(\mathcal{B}))}$. For the upper bound, we know $\frac{1}{n^3} \leq \frac{\varepsilon_1}{n(w(\mathcal{B})+h(\mathcal{B}))}$, and since $(1+\varepsilon_2)^2 \leq 1+\varepsilon_1$, we can replace the upper bound with $\frac{(1+\varepsilon_1)\texttt{uloss}(\mathcal{B})+\varepsilon_1/n}{w(\mathcal{B})+h(\mathcal{B})}$. Therefore, $\mathbf{ULI}_1(\mathcal{B})$ is a $(\frac{\varepsilon_1}{1+\varepsilon_1}\frac{\texttt{uloss}(\mathcal{B})}{w(\mathcal{B})+h(\mathcal{B})}, \varepsilon_1\frac{\texttt{uloss}(\mathcal{B})+1/n}{w(\mathcal{B})+h(\mathcal{B})})$-quality $\texttt{uloss}$-indicator for $\mathcal{B}$.

To analyze the running time, we see that $\mathbf{ULI}_1(\mathcal{B})$ consists of running $\mathbf{UGT}_1(\mathcal{B}, a, b)$ for different values of a, b until it outputs 0 or the loop stops. For any iteration of the loop, a will be $(1+\varepsilon_2)^i$ for some i and b will be $\frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^j}$ for some j. When $b = \frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^j}$, the loop will have reached this step iff it has not returned 0 at any previous point when the majority answer on line 2a was SMALL. This will have happened with probability $\frac{1}{(1+\varepsilon_2)^j}$, and if it did happen, the cost of the iteration would be $\widetilde{O}_{\varepsilon_2}(\frac{w(\mathcal{B})\sqrt{(1+\varepsilon_2)^i}}{\frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^j}})$. This yields a total of $\widetilde{O}_{\varepsilon_2}(\sum_{i,j}\frac{1}{(1+\varepsilon_2)^j}\frac{(w(\mathcal{B})+h(\mathcal{B}))\sqrt{(1+\varepsilon_2)^i}}{\frac{w(\mathcal{B})+h(\mathcal{B})}{(1+\varepsilon_2)^j}})$, where the sum is taken over all iterations of the loop.

Looking more closely at the procedure, if a ever exceeds $\texttt{uloss}(\mathcal{B})$, the calls to $\mathbf{UGT}_1$ will continue to return SMALL, meaning that i will never increase beyond $\log_{1+\varepsilon_2}(\texttt{uloss}(\mathcal{B}))$. Additionally, each run of the loop makes a and b closer by a factor of $1+\varepsilon_2$, so the number of iterations of the loop will be at most polylogarithmic in $w(\mathcal{B})+h(\mathcal{B})$. Furthermore, we note that if $\texttt{uloss}(\mathcal{B}) = 0$, the loop will still run with $a = 1$, so for this case, it is necessary to use 0 as the upper bound for the value of i instead of $\log_{1+\varepsilon_2}(\texttt{uloss}(\mathcal{B}))$. Simplifying, we get that the expected running time is $\widetilde{O}(\sqrt{\texttt{uloss}(\mathcal{B})})$ in the case $\texttt{uloss}(\mathcal{B}) > 0$ and $\widetilde{O}(1)$ in the case $\texttt{uloss}(\mathcal{B}) = 0$. Therefore, we can bound the running time of $\mathbf{ULI}_1(\mathcal{B})$ by $\widetilde{O}(\sqrt{\texttt{uloss}(\mathcal{B})}+1)$. □

We now prove Lemma 8.5.

Proof. We aim to show that $\mathbf{UGT}_1(\mathcal{B}, a, b)$ is a gap test for $\texttt{uloss}(\mathcal{B})$. We build a gap test from $\mathbf{XLI}_2$, which is a gap test for $\texttt{Xloss}(\mathcal{B})$ with an additive term. In order to handle the additive term, we'll need to go through some careful analysis and changes of parameters, allowing us to absorb the additive term into the multiplicative term in the process of converting the gap test into a gap test for $\texttt{uloss}(\mathcal{B})$. Intuitively, we are able to do this because the extra additive term depends on $h(\mathcal{B}) - w(\mathcal{B})$, and the process of converting from $\texttt{Xloss}(\mathcal{B})$ to $\texttt{uloss}(\mathcal{B})$ involves adding $h(\mathcal{B}) - w(\mathcal{B})$ after multiplying by 2. It is in this step of adding $h(\mathcal{B}) - w(\mathcal{B})$ that we "leave some space" for the additive error term.

First, note that $\mathcal{B}, a', r$ satisfy the input constraints of $\mathbf{XLI}_2(\mathcal{B}, a', r)$ in every call to $\mathbf{XLI}_2$ made by $\mathbf{UGT}_1(\mathcal{B}, a, b)$. Therefore by the guarantees of the algorithm, if $a' > \texttt{uloss}(\mathcal{B})$, then $\mathbf{XLI}_2(\mathcal{B}, a', r)$ is a $(5\varepsilon_3\frac{\texttt{Xloss}(\mathcal{B})}{w(\texttt{extL}_r(\mathcal{B}))}, 5\varepsilon_3\frac{\texttt{Xloss}(\mathcal{B})+1}{w(\texttt{extL}_r(\mathcal{B}))} + 2\varepsilon_3\frac{h(\mathcal{B})-w(\mathcal{B})}{w(\texttt{extL}_r(\mathcal{B}))})$-quality $\texttt{Xloss}$-indicator for $\mathcal{B}$. If $a' \leq \texttt{uloss}(\mathcal{B})$ we only know that $\mathbf{XLI}_2(\mathcal{B}, a', r)$ is a $(5\varepsilon_3\frac{\texttt{Xloss}(\mathcal{B})}{w(\texttt{extL}_r(\mathcal{B}))}, 1)$-quality $\texttt{Xloss}$-indicator for $\mathcal{B}$.

We note that $\texttt{Xloss}(\mathcal{B}) = w(\mathcal{B}) - \texttt{lcs}(\mathcal{B}) = (\texttt{uloss}(\mathcal{B}) + w(\mathcal{B}) - h(\mathcal{B}))/2$.

Let z be the sum of the outputs of the k calls to $\mathbf{XLI}_2(\mathcal{B}, a', r)$. We must show two things:

1. If $\texttt{uloss}(\mathcal{B}) < a$ then $\mathbf{UGT}_1$ returns BIG with probability at most 1/3.
2. If $\texttt{uloss}(\mathcal{B}) > b$ then $\mathbf{UGT}_1$ returns SMALL with probability at most 1/3.

We first consider the case that the algorithm returns BIG in step 1. In this case $a < b < (1+\varepsilon)(h(B) - w(B)) \leq (1+\varepsilon)\texttt{uloss}(\mathcal{B})$ which means that the hypothesis of part 1 fails and so part 1 vacuously true. Also part 2 is trivially true.

So we may assume that the algorithm does not return BIG in step 1 and therefore $b \geq (1+\varepsilon_3)(h(B) - w(B))$ which implies $(b - h(B) - w(B)) \geq \varepsilon_3 b$.

Let $\Theta = (1-6\varepsilon_3)\frac{k}{w(\texttt{extL}_r(\mathcal{B}))}\frac{b+w(\mathcal{B})-h(\mathcal{B})}{2}$. The algorithm returns BIG if $z > \Theta$ and returns SMALL if $z \leq \Theta$.

By the previous inequality and the choice of k in the algorithm we have:

$$\begin{aligned}\Theta &\geq (1-6\varepsilon_3)\frac{\varepsilon_3 bk}{2w(\texttt{extL}_r(\mathcal{B}))} \\ &\geq \frac{6}{(\varepsilon_3)^2}.\end{aligned}$$

To prove part 1, assume $\texttt{uloss}(\mathcal{B}) < a$.

Hence, $\texttt{uloss}(\mathcal{B}) < a'$ and by the correctness of $\mathbf{XLI}_2$,

$$\mathbb{E}[z] \leq k/w(\texttt{extL}_r(\mathcal{B})))[(1+5\varepsilon_3)\texttt{Xloss}(\mathcal{B}) + 5\varepsilon_3 + 2\varepsilon_3(h(\mathcal{B}) - w(\mathcal{B}))].$$

We claim:

$$(1+5\varepsilon_3)\texttt{Xloss}(\mathcal{B}) + 5\varepsilon_3 + 2\varepsilon_3(h(\mathcal{B}) - w(\mathcal{B})) \leq (1-20\varepsilon_3)(b + w(\mathcal{B}) - h(\mathcal{B}))/2. \tag{9.1}$$

To establish this claim, note first that $\texttt{Xloss}(\mathcal{B}) = (\texttt{uloss}(\mathcal{B}) + w(\mathcal{B}) - h(\mathcal{B}))/2 \leq (a + w(\mathcal{B}) - h(\mathcal{B}))/2$. Also by the input conditions $a \leq b/(1+\varepsilon_2) = b/(1+68\varepsilon_3) \leq b(1-54\varepsilon_3)$ since $\varepsilon_3 \leq \varepsilon_2/68 \leq 1/(68\times 4)$. Therefore $\texttt{Xloss}(\mathcal{B}) \leq (b + w(\mathcal{B}) - h(\mathcal{B}))/2 - \varepsilon_3(27b)$ and so the lefthand side of (9.1) is at most:

$$\begin{aligned} & \frac{b + w(\mathcal{B}) - h(\mathcal{B})}{2} + \varepsilon_3(-27b + 5 + 2(h(\mathcal{B}) - w(\mathcal{B})) \\ \leq\ & \frac{b + w(\mathcal{B}) - h(\mathcal{B})}{2} + \varepsilon_3(-22b + 2(h(\mathcal{B}) - w(\mathcal{B}))) \\ \leq\ & \frac{b + w(\mathcal{B}) - h(\mathcal{B})}{2} + \varepsilon_3(-10b - 10(h(\mathcal{B}) - w(\mathcal{B}))) \\ \leq\ & (1-20\varepsilon_3)\frac{b + w(\mathcal{B}) - h(\mathcal{B})}{2}, \end{aligned}$$

where the first inequality uses $b \geq 1$ and the second inequality uses $b \geq a \geq \texttt{uloss}(\mathcal{B}) \geq h(\mathcal{B}) - w(\mathcal{B})$.

We therefore have $z \leq (1-20\varepsilon_3)(b + w(\mathcal{B}) - h(\mathcal{B}))/2 \leq (1-14\varepsilon_3)\Theta$, where Θ was defined earlier. By the Chernoff bound in Theorem 2.1, the probability that $z > \Theta$ is at most $e^{-(14\varepsilon_3)^2\Theta}$. Using the upper bound $\Theta \geq \frac{6}{(\varepsilon_3)^2}$ stated earlier this is less than $1/3$.

Now we prove part 2. Assume $\texttt{uloss}(\mathcal{B}) > b$. Regardless of whether $a' > \texttt{uloss}(\mathcal{B})$, the expected sum of outputs is at least $(1-5\varepsilon_3)k\texttt{Xloss}(\mathcal{B})/w(\texttt{extL}_r(\mathcal{B}))$. Since $\texttt{Xloss}(\mathcal{B}) \geq (b + w(\mathcal{B}) - h(\mathcal{B}))/2$. the expected sum is at least $(1-5\varepsilon_3)(k/w(\texttt{extL}_r(\mathcal{B})))(b + w(\mathcal{B}) - h(\mathcal{B}))/2 \geq (1+\varepsilon_3)\Theta$. Again applying Theorem 2.1, the probability that $z \leq \Theta$ is at most $e^{-(\varepsilon_3)^2\Theta/3}$. By the inequality $\Theta \geq \frac{6}{(\varepsilon_3)^2}$, this is at most $e^{-2} \leq 1/3$.

To analyze the running time, $\mathbf{UGT}_1(\mathcal{B}, a, b)$ makes $\widetilde{O}(\frac{w(\mathcal{B})}{b})$ calls to $\mathbf{XLI}_2(\mathcal{B}, a', r)$ with $r = \widetilde{O}_{\varepsilon_2}(a)$. By assumption, $\mathbf{XLI}_2(\mathcal{B}, a', r)$ runs in time $\widetilde{O}_{\varepsilon_3}(\sqrt{r})$. Therefore, $\mathbf{UGT}_1(\mathcal{B}, a, b)$ runs in time $\widetilde{O}_{\varepsilon_2}(\frac{w(\mathcal{B})}{b}\sqrt{a})$. □

In order to prove Lemma 8.4, we will need the following proposition.

PROPOSITION 9.1. *Let $\mathcal{B}$ be a box, and let (x, y) be a point on the LCS of $\mathcal{B}$. Then $|(y - y_B(\mathcal{B}) + 1) - (x - x_L(\mathcal{B}))| \leq \texttt{uloss}(\mathcal{B})$.*

Proof. Let $\mathcal{B}_1$ be the box $(x_L(\mathcal{B}), x] \times [y_B(\mathcal{B}), y]$, and let $\mathcal{B}_2$ be the box $(x, x_R(\mathcal{B})] \times [y, y_T(\mathcal{B})]$. Since (x, y) is on the LCS of $\mathcal{B}$, $\texttt{lcs}(\mathcal{B}_1) + \texttt{lcs}(\mathcal{B}_2) = \texttt{lcs}(\mathcal{B})$, so $\texttt{uloss}(\mathcal{B}_1) + \texttt{uloss}(\mathcal{B}_2) = \texttt{uloss}(\mathcal{B})$, in particular, $\texttt{uloss}(\mathcal{B}_1) \leq \texttt{uloss}(\mathcal{B})$. Furthermore, $\texttt{lcs}(\mathcal{B}_1) \leq d_{min}(\mathcal{B}_1)$, so $h(\mathcal{B}_1) - w(\mathcal{B}_1) \leq w(\mathcal{B}_1) + h(\mathcal{B}_1) - 2 \cdot \texttt{lcs}(\mathcal{B}_1) \leq \texttt{uloss}(\mathcal{B})$. As a result, $(y - y_B(\mathcal{B}) + 1) - (x - x_L(\mathcal{B})) \leq \texttt{uloss}(\mathcal{B})$. Similarly, applying $\texttt{lcs}(\mathcal{B}_1) \leq h(\mathcal{B}_1)$ yields $(x - x_L(\mathcal{B})) - (y - y_B(\mathcal{B}) + 1) \leq \texttt{uloss}(\mathcal{B})$. □

We now prove Lemma 8.4.

Proof. Suppose that $\mathcal{B}, a', r$ satisfy the input constraints of $\mathbf{XLI}_1$ in Tab. 1. For convenience, let $\mathcal{B}' = extL_r(\mathcal{B})$. Since $2a' \leq r$, for any call $\mathbf{XLI}_1(\mathcal{T})$ made by $\mathbf{XLI}_2(\mathcal{B}, a', r)$, $\mathcal{T}$ satisfies the input constraints of $\mathbf{BGT}$. Therefore by assumption, $\mathbf{XLI}_1(\mathcal{T})$ is a $(\frac{2\varepsilon_3}{1+2\varepsilon_3}\frac{\texttt{Xloss}(\mathcal{T})}{w(\mathcal{T})}, \varepsilon_3\frac{\texttt{Xloss}(\mathcal{T})+1/n}{w(\mathcal{T})} + (1+\varepsilon_3)\frac{\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})})$-quality $\texttt{Xloss}$-indicator for $\mathcal{T}$. We aim to show that $\mathbf{XLI}_2(\mathcal{B}, a', r)$ is a $(5\varepsilon_3\frac{\texttt{Xloss}(\mathcal{B})}{w(\mathcal{B}')}, 5\varepsilon_3\frac{\texttt{Xloss}(\mathcal{B})+1}{w(\mathcal{B}')} + 2\varepsilon_3\frac{h(\mathcal{B})-w(\mathcal{B})}{w(\mathcal{B}')})$-quality $\texttt{Xloss}$-indicator for $\mathcal{B}$.

Partition $X(\texttt{extL}_r(\mathcal{B}))$ into intervals of size r, and let I_s be the set of intervals obtained by shifting this partition s indices to the right for $s \in [r]$. Let $\vec{\mathcal{T}}_s = \{I \times [x_L(I) - a', x_R(I) + a'] : I \in I_s\}$. We can think of the box $\mathcal{T}$ chosen by $\mathbf{XLI}_2$ as chosen by picking s uniformly at random, and then picking an element of $\vec{\mathcal{T}}_s$ uniformly at random.

Fix s, and let $J_1, J_2, \cdots, J_m$ be the elements of I_s. Let $\mathcal{T}_i$ be the box $J_i \times [x_L(J_i) - a', x_R(J_i) + a']$ (as in $\mathbf{XLI}_2$), and let S_i be the sequence achieving the maximum in $\texttt{terr}_{\varepsilon_3}(\mathcal{T}_i)$. Let $L_s = \bigcup_{\mathcal{T}_i \in \vec{\mathcal{T}}_s} S_i$. Since $y_T(\mathcal{T}_i) > y_B(\mathcal{T}_{i+1})$, it is possible that L_s is not monotone. Let L'_s be the longest common subsequence of L_s, i.e. L'_s is the resulting sequence after deleting as few points as possible from L_s to make it monotone. Let $S'_i = L'_s \cap \mathcal{T}_i$. Note that $S'_i \subseteq S_i$.

Let $\texttt{over}(L_s) = |L_s| - |L'_s|$, i.e. $\texttt{over}(L_s)$ is the number of points of L_s that must be deleted in order to make L_s monotone. We have

$$\begin{aligned} \texttt{over}(L_s) + \sum_{\mathcal{T}_i \in \vec{\mathcal{T}}_s} \texttt{Xloss}(S_i) &= \texttt{over}(L_s) + \texttt{Xloss}(L_s) \\ &= \texttt{Xloss}(L'_s) \geq \texttt{Xloss}(\mathcal{B}) \end{aligned}$$

so we aim to bound $\texttt{over}(L_s)$. In particular, we would

like to show

$$\sum_{s\in[r]} \texttt{over}(L_s) \le \varepsilon_3 \sum_{s\in[r]} \texttt{Xloss}(\mathcal{B}) = \varepsilon_3 r \texttt{Xloss}(\mathcal{B}) \tag{9.2}$$

By the quality guarantee of $\mathbf{XLI}_1$, $\mathbf{XLI}_1(\mathcal{T})$ returns 1 with probability at least

$$\frac{\texttt{Xloss}(\mathcal{T})}{w(\mathcal{T})} - \frac{2\varepsilon_3}{1+2\varepsilon_3}\frac{\texttt{Xloss}(\mathcal{T})}{w(\mathcal{T})} = \frac{1}{1+2\varepsilon_3}\frac{\texttt{Xloss}(\mathcal{T})}{w(\mathcal{T})}$$

Additionally, by definition of S_i, $\texttt{Xloss}(S_i) \le (1+\varepsilon_3)\texttt{Xloss}(\mathcal{T}_i)$. Hence, $\mathbf{XLI}_2(\mathcal{B}, a', r)$ returns 1 with probability at least

$$\begin{aligned}
&\sum_{s\in[r]}\sum_{\mathcal{T}_i\in\vec{\mathcal{T}}_s} \frac{1}{w(\mathcal{B}')}\frac{1}{1+2\varepsilon_3}\frac{\texttt{Xloss}(\mathcal{T}_i)}{w(\mathcal{T}_i)} \\
&\ge \sum_{s\in[r]}\sum_{\mathcal{T}_i\in\vec{\mathcal{T}}_s} \frac{1}{(1+2\varepsilon_3)(1+\varepsilon_3)}\frac{\texttt{Xloss}(S_i)}{rw(\mathcal{B}')} \\
&\ge \sum_{s\in[r]} \frac{1}{(1+2\varepsilon_3)(1+\varepsilon_3)}\frac{\texttt{Xloss}(\mathcal{B}) - \texttt{over}(L_s)}{rw(\mathcal{B}')} \\
&\ge \frac{1}{(1+2\varepsilon_3)(1+\varepsilon_3)}\frac{r\texttt{Xloss}(\mathcal{B}) - \varepsilon_3 r\texttt{Xloss}(\mathcal{B})}{rw(\mathcal{B}')} \\
&\ge \frac{1-\varepsilon_3}{(1+2\varepsilon_3)(1+\varepsilon_3)}\frac{\texttt{Xloss}(\mathcal{B})}{w(\mathcal{B}')} \ge (1-5\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{B})}{w(\mathcal{B}')}
\end{aligned}$$

Therefore, we focus on establishing 9.2.

We will characterize the points contributing to $\texttt{over}(L_s)$ as red and purple. More formally, we partition these points into two sets, $\texttt{RED}_s$ and $\texttt{PURPLE}_s$. Note that any point P on S_i contributing to $\texttt{over}(L_s)$ will be in violation with one or more points on S_{i-1} or S_{i+1} (but not both). If for every such point Q in violation with P, $|x(Q)-x(P)| \le \frac{8a'}{\varepsilon_3}$, then we label P as red ($P \in \texttt{RED}_s$). Otherwise, we label P as purple ($P \in \texttt{PURPLE}_s$). We will show that for any s, the number of purple points is small relative to $\texttt{Xloss}(\mathcal{B})$. The same statement will not hold for red points, but we will be able to bound the total number of red points when summing over all values of s. Putting these two statements together will give us 9.2. We start by focusing on purple points.

For a fixed s, let P_1 be a purple point contributing to $\texttt{over}(L_s)$. Let i be such that P_1 lies on S_i. By definition, there exists a point P_2 on either S_{i-1} or S_{i+1} in violation with P_1 such that $|x(P_1) - x(P_2)| > \frac{8a'}{\varepsilon_3}$. WLOG P_2 lies on S_{i+1}. Since $|Y(\mathcal{T}_i) \cap Y(\mathcal{T}_{i+1})| = 2a'+1$ and $y(P_1) > y_B(\mathcal{T}_{i+1}), y(P_2) < y_T(\mathcal{T}_i)$, at most $2a'-1$ of the at least $\frac{8a'}{\varepsilon_3}$ X-indices between P_1 and P_2 can have matches on L_s. Therefore, $\texttt{Xloss}(S_i) + \texttt{Xloss}(S_{i+1}) \ge \frac{8a'}{\varepsilon_3} - 2a' \ge \frac{6a'}{\varepsilon_3}$. Furthermore, since again $|Y(\mathcal{T}_i) \cap Y(\mathcal{T}_{i+1})| = 2a'+1$, we can always delete at most a' points to remove all violations between $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$, so $\texttt{over}_i(L_s) \le a'$. Summing over all $\mathcal{T}_i \in \vec{\mathcal{T}}_s$, the total number of purple points contributing to $\texttt{over}(L_s)$ is at most $\frac{\varepsilon_3}{3}\sum_i \texttt{Xloss}(S_i) \le \frac{\varepsilon_3}{3}(1+\varepsilon_3)\texttt{Xloss}(\mathcal{B}) \le \frac{2\varepsilon_3}{3}\texttt{Xloss}(\mathcal{B})$.

To analyze the red points, it will be helpful to look at the following picture. For a fixed s, consider two consecutive boxes $\mathcal{T}_i, \mathcal{T}_{i+1} \in \vec{\mathcal{T}}_s$. We would like to look at the region containing the red points that come from the pair $\mathcal{T}_i, \mathcal{T}_{i+1}$. Let $\mathcal{R}_i$ be the box $[x_R(\mathcal{T}_i) - \frac{8a'}{\varepsilon_3}, x_L(\mathcal{T}_{i+1}) + \frac{8a'}{\varepsilon_3}] \times [y_B(\mathcal{T}_{i+1}), y_T(\mathcal{T}_i)]$. Note that every red point P in $\mathcal{T}_i$ in violation with some point on S_{i+1} lies inside $\mathcal{R}_i$. Furthermore, every point on S_{i+1} in violation with P also lies in $\mathcal{R}_i$. Therefore, if we let $\vec{\mathcal{R}}_s$ be the set of all $\mathcal{R}_i$ corresponding to some $\mathcal{T}_i \in \vec{\mathcal{T}}_s$, the set of all red points as well as the set of all points on L_s in violation with them are both contained in $\vec{\mathcal{R}}_s$.

We will consider the union of $\vec{\mathcal{R}}_s$ for several different choices of s. Pictorally, $\vec{\mathcal{R}}_s$ consists of a sequence of boxes along the main diagonal of $\mathcal{B}$, forming a staircase with vertical gaps between successive stairs. For different choices of s, we will get such staircases offset by different amounts. In particular, if we look at $\vec{\mathcal{R}}_s$ and $\vec{\mathcal{R}}_{s+2a'+1}$, the two boxes $\mathcal{R}_i \in \vec{\mathcal{R}}_s$ and $\mathcal{R}'_i \in \vec{\mathcal{R}}_{s+2a'+1}$ corresponding to the same value i will lie one on top of the other. They will be disjoint, but there will be no vertical space between them, i.e. $y_T(\mathcal{R}_i) = y_B(\mathcal{R}'_i) - 1$.

The choices of s we include in our union will be an arithmetic sequence $\{\alpha + (2a'+1)\beta : \beta \in [\frac{r}{2a'+1} - 1]\}$ for some choice of $\alpha \in [2a']$. For such an α, let Q_α be the union of $\vec{\mathcal{R}}_s$ for each s in the arithmetic sequence corresponding to α, i.e. Q_α consists of the union of $\vec{\mathcal{R}}_\alpha, \vec{\mathcal{R}}_{\alpha+2a'+1}, \cdots$. Notice that all of the boxes in Q_α are disjoint, and there is no vertical space between any two consecutive boxes.

For a fixed α, consider a box $\mathcal{R}_i$ in Q_α, and let s be such that $\mathcal{R}_i \in \vec{\mathcal{R}}_s$. By construction, for any $s' \ne s$, $Y(\mathcal{R}_i)$ is disjoint from $Y(\mathcal{R}')$ for each $\mathcal{R}' \in \vec{\mathcal{R}}_{s'}$. Therefore, $\mathcal{R}_i$ intersects at most one (and as it turns out, exactly one) box in $\vec{\mathcal{T}}_{s'}$. This means that none of the points in $\mathcal{R}_i$ contribute to $\texttt{over}(L_s)$, i.e. their intersections with each of $\texttt{RED}_{s'}$ and $\texttt{PURPLE}_{s'}$ are empty. As a result, each red point lying in Q_α is in exactly one such $\texttt{RED}s$.

Now consider a fixed LCS $S_\mathcal{B}$ of $\mathcal{B}$. In each box $\mathcal{R}_i \in Q_\alpha$, there may be some points that do not lie on $S_\mathcal{B}$. Each such point represents a distinct X-index that contributes to $\texttt{Xloss}(\mathcal{B})$, since the input sequences were nonrepeating. Therefore, the total number of points inside all of the boxes $\mathcal{R}_i \in Q_\alpha$ that do not lie on $S_\mathcal{B}$ is a lower bound for $\texttt{Xloss}(\mathcal{B})$.

Recall that the red points in $\mathcal{R}_i$ were chosen by finding the smallest set of points that we needed to delete from S_i and S_{i+1} to remove all violations. Since the points lying on $S_{\mathcal{B}}$ form a common subsequence, the number of points in $\mathcal{R}_i$ that do not lie on $S_{\mathcal{B}}$ is an upper bound for the number of red points in $\mathcal{R}_i$. Applying this argument to all boxes $\mathcal{R}_i$ and using the conclusion from the previous paragraph, it follows that the total number of red points in all boxes $\mathcal{R}_i \in Q_\alpha$ is at most $\texttt{Xloss}(\mathcal{B})$.

Putting things together,

$$\sum_{s\in[r]} |\texttt{RED}_s| = \sum_{\alpha} \sum_{\vec{\mathcal{R}}_s \in Q_\alpha} |\texttt{RED}_s| \le \sum_{\alpha} \texttt{Xloss}(\mathcal{B})$$
$$= \frac{2a'+1}{r} \sum_{s} \texttt{Xloss}(\mathcal{B}) \le \frac{\varepsilon_3}{3} \sum_{s} \texttt{Xloss}(\mathcal{B})$$

Combining this with the contribution from $\texttt{PURPLE}_s$, $\sum_{s\in[r]} \texttt{over}(L_s) \le \varepsilon_3 \sum_{s\in[r]} \texttt{Xloss}(\mathcal{B})$, establishing 9.2 as desired.

Next, assuming that $a' > \texttt{uloss}(\mathcal{B})$, we aim to upper bound the probability that $\mathbf{XLI}_2$ returns 1. In particular, we would like to show this probability to be at most

$$(1+5\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{B})}{w(\mathcal{B}')} + 2\varepsilon_3 \frac{h(\mathcal{B})-w(\mathcal{B})}{w(\mathcal{B}')} + 5\varepsilon_3 \frac{1}{w(\mathcal{B}')}$$

By the quality guarantee of $\mathbf{XLI}_1$, $\mathbf{XLI}_1(\mathcal{T})$ returns 1 with probability at most

$$\frac{\texttt{Xloss}(\mathcal{T})}{w(\mathcal{T})} + \varepsilon_3 \frac{\texttt{Xloss}(\mathcal{T})+1/n}{w(\mathcal{T})} + (1+\varepsilon_3)\frac{\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})}$$
$$= (1+\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{T})}{w(\mathcal{T})} + (1+\varepsilon_3)\frac{\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})} + \frac{\varepsilon_3}{nw(\mathcal{T})}$$

Therefore, $\mathbf{XLI}_2(\mathcal{B}, a', r)$ returns 1 with probability at most

$$\sum_{s\in[r]} \sum_{\mathcal{T}_i \in \vec{\mathcal{T}}_s} \frac{1}{w(\mathcal{B}')}\Big((1+\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{T}_i)}{w(\mathcal{T}_i)} + (1+\varepsilon_3)\frac{\texttt{terr}_{\varepsilon_3}(\mathcal{T}_i)}{w(\mathcal{T}_i)} + \frac{\varepsilon_3}{nw(\mathcal{T}_i)}\Big)$$

We will need to upper bound the $\texttt{terr}_{\varepsilon_3}(\mathcal{T})$ term. If we can show

$$\sum_{s\in[r]} \sum_{\mathcal{T}_i \in \vec{\mathcal{T}}_s} \frac{\texttt{terr}_{\varepsilon_3}(\mathcal{T}_i)}{w(\mathcal{T}_i)}$$
$$\le \sum_{s\in[r]} \frac{\varepsilon_3(4+\varepsilon_3)\texttt{Xloss}(\mathcal{B}) + \varepsilon_3(h(\mathcal{B})-w(\mathcal{B}))}{r} \tag{9.3}$$

we will be able to achieve our desired result. We will also need

$$\sum_{i=1}^{m} \texttt{Xloss}(\mathcal{T}_i) \le \texttt{Xloss}(\mathcal{B}) \tag{9.4}$$

Using 9.3 and 9.4, the probability that $\mathbf{XLI}_2(\mathcal{B}, a', r)$ returns 1 is at most

$$\sum_{s\in[r]} \sum_{\mathcal{T}_i \in \vec{\mathcal{T}}_s} \frac{1}{w(\mathcal{B}')}\Big((1+\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{T}_i)}{w(\mathcal{T}_i)} +$$
$$(1+\varepsilon_3)\frac{\texttt{terr}_{\varepsilon_3}(\mathcal{T}_i)}{w(\mathcal{T}_i)} + \frac{\varepsilon_3}{nw(\mathcal{T}_i)}\Big)$$
$$\le \Big(\sum_{s\in[r]} \frac{1+\varepsilon_3}{w(\mathcal{B}')}\frac{\texttt{Xloss}(\mathcal{B})}{r}\Big) + \frac{\varepsilon_3}{w(\mathcal{B}')} +$$
$$\frac{1+\varepsilon_3}{w(\mathcal{B}')} \sum_{s\in[r]} \frac{\varepsilon_3(4+\varepsilon_3)\texttt{Xloss}(\mathcal{B}) + \varepsilon_3(h(\mathcal{B})-w(\mathcal{B}))}{r}$$
$$\le (1+\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{B})}{w(\mathcal{B}')} + \frac{\varepsilon_3(1+\varepsilon_3)(4+\varepsilon_3)\texttt{Xloss}(\mathcal{B})}{w(\mathcal{B}')} +$$
$$\frac{\varepsilon_3(1+\varepsilon_3)(h(\mathcal{B})-w(\mathcal{B}))}{w(\mathcal{B}')} + \frac{\varepsilon_3}{w(\mathcal{B}')}$$
$$\le (1+5\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{B})}{w(\mathcal{B}')} + 2\varepsilon_3\frac{h(\mathcal{B})-w(\mathcal{B})}{w(\mathcal{B}')} + 5\varepsilon_3\frac{1}{w(\mathcal{B}')}$$

Hence, we focus on establishing 9.3 and 9.4.

To show 9.4, note that $X(\mathcal{B}) \subseteq \bigcup J_i$. Since $r \ge a' \ge \texttt{uloss}(\mathcal{B})$, by Prop. 9.1, for any point P on the LCS of $\mathcal{B}$, if $x(P) \in J_i$, then $y(P) \in Y(\mathcal{T}_i)$. This means that $\texttt{lcs}(\mathcal{B}) \cap J_i|\mathcal{B} \subseteq \mathcal{T}_i$, so the LCS of each $\mathcal{T}_i$ is at least as long as the portion of $\texttt{lcs}(\mathcal{B})$ lying in $J_i|\mathcal{B}$. Since the points in $\mathcal{T}_1, \mathcal{T}_m$ which lie outside of $\mathcal{B}$ form common subsequences outside the Y range of $\mathcal{B}$, they do not contribute to $\texttt{Xloss}(\mathcal{T}_1)$, $\texttt{Xloss}(\mathcal{T}_m)$, respectively, so $\sum_{i=1}^{m} \texttt{Xloss}(\mathcal{T}_i) \le \texttt{Xloss}(\mathcal{B})$ as desired.

To show 9.3, we need to look at $\texttt{terr}_{\varepsilon_3}(\mathcal{T}_i)$. Let S_i, S_i' be as defined above, so

$$\texttt{terr}_{\varepsilon_3}(\mathcal{T}_i) = \texttt{serr}_{\varepsilon_3}(S_i, \mathcal{T}_i)$$
$$\le \texttt{herr}(S_i', \mathcal{T}_i) + \varepsilon_3 \texttt{Yloss}_{\texttt{trim}}(S_i', \mathcal{T}_i)$$

Since L_s' is monotone, $\sum_{i=1}^{m} \texttt{Yloss}_{\texttt{trim}}(S_i', \mathcal{T}_i)$ counts each Y-index not on L_s' at most once, so

$$\varepsilon_3 \sum_{i=1}^{m} \texttt{Yloss}_{\texttt{trim}}(S_i', \mathcal{T}_i) \le$$
$$\varepsilon_3(h(\mathcal{B}) - w(\mathcal{B}) + \sum_{i=1}^{m} \texttt{Xloss}(S_i', \mathcal{T}_i))$$

This gives us

$$\sum_{i=1}^{m} \texttt{herr}(S_i', \mathcal{T}_i) + \varepsilon_3 \texttt{Yloss}_{\texttt{trim}}(S_i', \mathcal{T}_i) \le \varepsilon_3(h(\mathcal{B}) - w(\mathcal{B})) +$$
$$\sum_{i=1}^{m} ((1+\varepsilon_3)\texttt{Xloss}(S_i', \mathcal{T}_i) - \texttt{Xloss}(\mathcal{T}_i))$$

We have

$$\sum_{i=1}^{m} \texttt{Xloss}(S_i', \mathcal{T}_i) = \texttt{over}(L_s) + \sum_{i=1}^{m} \texttt{Xloss}(S_i, \mathcal{T}_i)$$
$$\leq \texttt{over}(L_s) + (1+\varepsilon_3) \sum_{i=1}^{m} \texttt{Xloss}(\mathcal{T}_i)$$

Therefore, using 9.2 and 9.4,

$$\sum_{s\in[r]} ((\sum_{i=1}^{m} (1+\varepsilon_3)\texttt{Xloss}(S_i', \mathcal{T}_i) - \texttt{Xloss}(\mathcal{T}_i)) +$$
$$\varepsilon_3(h(\mathcal{B}) - w(\mathcal{B})))$$
$$\leq \sum_{s\in[r]} ((1+\varepsilon_3)(\texttt{over}(L_s) + (1+\varepsilon_3)\sum_{i=1}^{m} \texttt{Xloss}(\mathcal{T}_i)) -$$
$$\sum_{i=1}^{m} \texttt{Xloss}(\mathcal{T}_i) + \varepsilon_3(h(\mathcal{B}) - w(\mathcal{B})))$$
$$\leq (1+\varepsilon_3)\varepsilon_3 \sum_{s\in[r]} \texttt{Xloss}(\mathcal{B}) + \sum_{s\in[r]} (2\varepsilon_3 + \varepsilon_3^2)\texttt{Xloss}(\mathcal{B}) +$$
$$\varepsilon_3 \sum_{s\in[r]} (h(\mathcal{B}) - w(\mathcal{B}))$$
$$\leq (\sum_{s\in[r]} 2\varepsilon_3 \texttt{Xloss}(\mathcal{B}) + (2\varepsilon_3 + \varepsilon_3^2)\texttt{Xloss}(\mathcal{B}) +$$
$$\varepsilon_3(h(\mathcal{B}) - w(\mathcal{B})))$$
$$\leq (\sum_{s\in[r]} \varepsilon_3(4+\varepsilon_3)\texttt{Xloss}(\mathcal{B}) + \varepsilon_3(h(\mathcal{B}) - w(\mathcal{B})))$$

Since $w(\mathcal{T}_i) = r$, we have 9.3 as desired.

Putting the two bounds together, $\mathbf{XLI}_2(\mathcal{B}, a', r)$ is a $(5\varepsilon_3 \frac{\texttt{Xloss}(\mathcal{B})}{w(\mathcal{B}')}, 5\varepsilon_3 \frac{\texttt{Xloss}(\mathcal{B})+1}{w(\mathcal{B}')} + 2\varepsilon_3 \frac{h(\mathcal{B})-w(\mathcal{B})}{w(\mathcal{B}')})$-quality $\texttt{Xloss}$-indicator for $\mathcal{B}$. The running time claim follows from the assumptions regarding $\mathbf{XLI}_1$, as $\mathbf{XLI}_2(\mathcal{B}, a', r)$ consists of running $\mathbf{XLI}_1$ on a box of size r. □

We now prove Lemma 8.3.

Proof. $\mathbf{XLI}_1(\mathcal{T})$ seeks to output 1 with probability approximately equal to $\texttt{Xloss}(\mathcal{T})/w(\mathcal{T})$, and 0 otherwise. The procedure involves a sequence of iterations. In each iteration, the procedure takes a majority vote of $O(\log n)$ instances of the gap test $\mathbf{BGT}$ with a particular lower threshold a and upper threshold b, which differ by a factor of $(1+\varepsilon_3)$. If the iteration returns BIG, the procedure outputs 1. If the iteration returns SMALL, the procedure may stop and return 0 with a small probability. Otherwise, a and b are decreased by a factor of $(1+\varepsilon_3)$, and these new values are used in the next iteration.

We say that each iteration is *reliable* if the majority of $O(\log n)$ runs of $\mathbf{BGT}$ is correct, and $\mathbf{XLI}_1(\mathcal{T})$ is *reliable* if all of the iterations are reliable. By a Chernoff bound, each iteration is unreliable with probability at most $\frac{1}{n^6}$. By a union bound, using the fact that the number of iterations is at most $O(\frac{1}{\varepsilon_3}\log n)$ the probability that the run of $\mathbf{XLI}_1$ is unreliable is at most $\frac{1}{\varepsilon_3 n^5}$. For $\varepsilon_3 \geq \frac{1}{n^2}$, this is at most $\frac{1}{n^3}$.

The correctness condition for $\mathbf{XLI}_1$ is that the probability that 1 is output is within certain bounds. We now compute the probability that the output is 1 conditioned on the run being reliable. We then have to make a slight adjustment because of the chance it is unreliable.

So assume that the run of $\mathbf{XLI}_1$ is reliable. Let j_1, j_2 be the largest and smallest nonnegative integers respectively such that $\frac{w(\mathcal{T})}{(1+\varepsilon_3)^{j_2}} < \texttt{Xloss}(\mathcal{T}) \leq \texttt{Xloss}(\mathcal{T}) + \texttt{terr}_{\varepsilon_3}(\mathcal{T}) \leq \frac{w(\mathcal{T})}{(1+\varepsilon_3)^{j_1}}$. In $\mathbf{XLI}_1(\mathcal{T})$, the test on line 2a will return SMALL for $i \leq j_1 - 1$, and it will return BIG for $i = j_2$. In order for $\mathbf{XLI}_1(\mathcal{T})$ to return 1, it has to not return 0 via line 2b on each of the first j_1 iterations of the loop. This happens with probability $\frac{1}{(1+\varepsilon_3)^{j_1}}$, so $\mathbf{XLI}_1(\mathcal{T})$ returns 1 with probability at most $\frac{1}{(1+\varepsilon_3)^{j_1}} \leq (1+\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{T})+\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})}$.

On the other hand, the test on line 2a will return BIG after at most j_2 iterations of the loop, meaning line 2b has a chance of returning 0 on at most j_2 iterations. The probability that it does not return 0 on any of these is at least $\frac{1}{(1+\varepsilon_3)^{j_2}}$, so $\mathbf{XLI}_1(\mathcal{T})$ returns 1 with probability at least $\frac{1}{(1+\varepsilon_3)^{j_2}} \geq \frac{\texttt{Xloss}(\mathcal{T})}{(1+\varepsilon_3)w(\mathcal{T})}$.

Combining this with the probability that the run is unreliable via a union bound, the probability that $\mathbf{XLI}_1(\mathcal{T})$ returns 1 is at least $\frac{\texttt{Xloss}(\mathcal{T})}{(1+\varepsilon_3)w(\mathcal{T})} - \frac{1}{n^3}$ and at most $(1+\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{T})+\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})} + \frac{1}{n^3}$. Looking more closely at the lower bound, if $\texttt{Xloss}(\mathcal{T}) = 0$, then since the probability of returing 1 is at least 0, it must be at least $\frac{\texttt{Xloss}(\mathcal{T})}{(1+2\varepsilon_3)w(\mathcal{T})}$. If instead $\texttt{Xloss}(\mathcal{T}) \geq 1$, then since $n \geq w(\mathcal{T})$, $\frac{1}{n^2} \leq \frac{\varepsilon_3}{(1+\varepsilon_3)(1+2\varepsilon_3)}$, $\frac{\texttt{Xloss}(\mathcal{T})}{(1+\varepsilon_3)w(\mathcal{T})} - \frac{1}{n^3} \leq \frac{\texttt{Xloss}(\mathcal{T})}{(1+2\varepsilon_3)w(\mathcal{T})}$. For the upper bound, since $\frac{1}{n^3} \leq \frac{\varepsilon_3}{nw(\mathcal{T})}$. $(1+\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{T})+\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})} + \frac{1}{n^3}$ is at most $(1+\varepsilon_3)\frac{\texttt{Xloss}(\mathcal{T})+\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})} + \frac{\varepsilon_3}{nw(\mathcal{T})}$. Therefore, $\mathbf{XLI}_1(\mathcal{T})$ is a $(\frac{2\varepsilon_3}{1+2\varepsilon_3}\frac{\texttt{Xloss}(\mathcal{T})}{w(\mathcal{T})}, \varepsilon_3\frac{\texttt{Xloss}(\mathcal{T})+1/n}{w(\mathcal{T})} + (1+\varepsilon_3)\frac{\texttt{terr}_{\varepsilon_3}(\mathcal{T})}{w(\mathcal{T})})$-quality $\texttt{Xloss}$ indicator for $\mathcal{T}$.

To analyze the running time, we see that $\mathbf{XLI}_1(\mathcal{T})$ consists of running $\mathbf{BGT}(\mathcal{T}, a, b)$ for different values of a, b with $b = (1+\varepsilon_3)a$ until it outputs 0 or 1. Once i exceeds the value of $\log_{1+\varepsilon_3}(\frac{w(\mathcal{T})}{\texttt{Xloss}(\mathcal{T})})$, $\frac{w(\mathcal{T})}{(1+\varepsilon_3)^i}$ will be smaller than $\texttt{Xloss}(\mathcal{T})$, meaning $\mathbf{BGT}(\mathcal{T}, \frac{w(\mathcal{T})}{(1+\varepsilon_3)^{i+1}}, \frac{w(\mathcal{T})}{(1+\varepsilon_3)^i})$ will return BIG and the loop

will stop. For i between 0 and $\log_{1+\varepsilon_3}(\frac{w(\mathcal{T})}{\texttt{Xloss}(\mathcal{T})})$, the loop will reach the i^{th} step if it has not returned 0 in any previous step. This happens with probability $\frac{1}{(1+\varepsilon_3)^i}$, and if it does happen, the cost of the i^{th} step will be $\tilde{O}((1+\varepsilon_3)^i\sqrt{w(\mathcal{T})})$. Summing over all i, the total expected running time is $\tilde{O}(\sqrt{w(\mathcal{T})})$. □

We now prove Lemma 8.2. We will need the following claim, which is a minor modification of Claim 4.1 in [5].

Claim 9.1. *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 $\Pr[|q/p - m| \geq \varepsilon_1 m + \frac{16(2e-1)\log n}{\varepsilon_1^2 p}] \leq \frac{1}{n^3}$*

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.

If $m \leq \frac{16\log n}{\varepsilon_1^2 p}$, then by Theorem 2.2,

$\Pr[|\frac{q}{p} - m| \geq m\frac{16(2e-1)\log n}{\varepsilon_1^2 pm}] \leq 2^{-(1+\frac{16(2e-1)\log n}{\varepsilon_1^2 pm})pm} \leq \frac{1}{n^3}$.

If $m > \frac{16\log n}{\varepsilon_1^2 p}$, then by Theorem 2.2,

$\Pr[|\frac{q}{p} - m| \geq \varepsilon_1 m] \leq 2e^{-\varepsilon_1^2 pm/4} \leq 2e^{-4\log n} \leq \frac{1}{n^3}$. □

We use Claim 9.1 to prove Lemma 8.2.

Proof. First, note that since $\mathbf{XLI}_0$ succeeds with probability 2/3, the majority taken on line 3 succeeds with probability $1 - n^{-\Omega(\log n)}$.

Let $d_{out}(\mathcal{T})$ be the number of indices in $X(\mathcal{T})$ whose match lies outside of $\mathcal{T}$. Let $d_{in}(\mathcal{T})$ be the number of indices in $X(\mathcal{T})$ whose match lies inside $\mathcal{T}$, but not on the LCS of $\mathcal{T}$ (Note that this definition is a rewording of the definition given in section 2). We have $d_{out}(\mathcal{T}) + d_{in}(\mathcal{T}) = \texttt{Xloss}(\mathcal{T})$. Our goal will be to show that V approximates $d_{out}(\mathcal{T})$ and W approximates $d_{in}(\mathcal{T})$.

We first show V approximates $d_{out}(\mathcal{T})$. Let N be the number of matches in $\mathcal{T}$. When $b \leq \frac{16(2e-1)}{\varepsilon_4^2}\sqrt{w(\mathcal{T})}\log n$, we get the exact value of N (and therefore $d_{out}(\mathcal{T})$) by reading the whole block. Now consider the case $b > \frac{16(2e-1)}{\varepsilon_4^2}\sqrt{w(\mathcal{T})}\log n$. As defined in **BGT**, let M be the number of collisions which lie in $\mathcal{T}$. Since each point has its X-index and Y-index each selected independently with probability p, we have $\mathbb{E}[M] = p^2 N$. Consider two cases (C' a constant).

If $N > \frac{2\varepsilon_4 b}{(2e-1)}$, then by Theorem 2.2,

$$\Pr[|\frac{M}{p^2} - N| \geq N\frac{2\varepsilon_4 b}{N}] \leq 2e^{-(\frac{2\varepsilon_4 b}{N})^2 p^2 \frac{N}{4}}$$
$$\leq 2e^{-\frac{\varepsilon_4^2 b^2}{N}\frac{C' w(\mathcal{T})\log^2 n}{b^2}} \leq 2e^{-C'\varepsilon_4^2\log^2 n} \leq \frac{1}{n^2}.$$

If $N \leq \frac{2\varepsilon_4 b}{(2e-1)}$, then by Theorem 2.2,

$$\Pr[|\frac{M}{p^2} - N| \geq N\frac{2\varepsilon_4 b}{N}] \leq 2^{-(1+\frac{2\varepsilon_4 b}{N})p^2 N}$$
$$\leq 2^{-(1+\frac{2\varepsilon_4 b}{N})*\frac{C' N w(\mathcal{T})\log^2 n}{b^2}} \leq 2^{-2C'\varepsilon_4\frac{w(\mathcal{T})}{b}\log^2 n} \leq \frac{1}{n^2}.$$

Thus, with high probability, $|\frac{M}{p^2} - N| < 2\varepsilon_4 b$, so $|V - d_{out}(\mathcal{T})| < 2\varepsilon_4 b$

Next, we show that W approximates $d_{in}(\mathcal{T})$. Let m be the number of matches in $\mathcal{T}$ that would be classified as bad by $\mathbf{XLI}_0$. By assumption,

$$d_{in}(\mathcal{T}) \leq m$$
$$\leq (1+\varepsilon_4)(\texttt{Xloss}(\mathcal{T}) + \texttt{terr}_{\varepsilon_3}(\mathcal{T})) - d_{out}(\mathcal{T})$$

Recall that in **BGT**, D is the number of sampled points classified as bad by $\mathbf{XLI}_0$. By Claim 9.1, with high probability,

$$(1-\varepsilon_4)m - \frac{16(2e-1)\log n}{\varepsilon_4^2 r^2} \leq \frac{D}{r^2}$$
$$\leq (1+\varepsilon_4)m + \frac{16(2e-1)\log n}{\varepsilon_4^2 r^2}$$

Thus, with high probability,

$$W - (\texttt{Xloss}(\mathcal{T}) + \texttt{terr}_{\varepsilon_3}(\mathcal{T}) - d_{out}(\mathcal{T}))$$
$$\leq \varepsilon_4((1+\varepsilon_4)(\texttt{Xloss}(\mathcal{T}) + \texttt{terr}_{\varepsilon_3}(\mathcal{T})) - d_{out}(\mathcal{T})) + \varepsilon_4 b$$

and

$$d_{in}(\mathcal{T}) - W \leq \varepsilon_4 d_{in}(\mathcal{T}) + \varepsilon_4 b$$

If $\texttt{Xloss}(\mathcal{T}) + \texttt{terr}_{\varepsilon_3}(\mathcal{T}) < a \leq \frac{b}{1+10\varepsilon_4}$, then

$$|(V+W) - (\texttt{Xloss}(\mathcal{T}) + \texttt{terr}_{\varepsilon_3}(\mathcal{T}))|$$
$$\leq |V - d_{out}(\mathcal{T})| + |W - (\texttt{Xloss}(\mathcal{T}) +$$
$$\texttt{terr}_{\varepsilon_3}(\mathcal{T}) - d_{out}(\mathcal{T}))|$$
$$\leq 2\varepsilon_4 b + \varepsilon_4(1+\varepsilon_4)b + \varepsilon_4 b \leq 5\varepsilon_4 b$$

Therefore, $V + W < (1-4\varepsilon_4)b$.

On the other hand, if $\texttt{Xloss}(\mathcal{T}) > b$, then

$$V + W \geq d_{out}(\mathcal{T}) - 2\varepsilon_4 b + (1-\varepsilon_4)d_{in}(\mathcal{T}) - \varepsilon_4 b$$
$$> (1-\varepsilon_4)b - 3\varepsilon_4 b \geq (1-4\varepsilon_4)b$$

Thus, the test on line 4 successfully distinguishes between these two cases.

To analyze the running time, first note that if $b = \tilde{O}(\sqrt{w(\mathcal{T})})$, then $\frac{w(\mathcal{T})\sqrt{w(\mathcal{T})}}{b} = \tilde{\Omega}(w(\mathcal{T}))$, meaning that if the condition on line 1a is met, the linear amount of time it takes to read each character in $X(\mathcal{T}), Y(\mathcal{T})$ is within this bound. The number of characters read on lines 1b and 2 is also seen to be within this bound with high probability by a standard Chernoff argument (for line 2 using the fact that $b \leq w(\mathcal{T})$. Furthermore, another Chernoff argument gives us that, with high probability, the number of collisions on line 2 is $\widetilde{O}(\frac{w(\mathcal{T})}{b})$. By assumption $\mathbf{XLI}_0$ runs in time $\widetilde{O}(\sqrt{w(\mathcal{T})})$, so line 3 runs in time $\widetilde{O}(\frac{w(\mathcal{T})\sqrt{w(\mathcal{T})}}{b})$. Therefore, all of these steps can be shown to run in time $\widetilde{O}(\frac{w(\mathcal{T})}{b}\sqrt{w(\mathcal{T})})$. □

10 Applying the LIS algorithm

Here we describe our use of the [16] algorithms, along with the improvements and modifications that we make. Due to space limitations, we omit the full details, instead providing a high level overview of this task. As such an overview was provided in 6, we focus primarily on our improvement to the LIS algorithm here.

As mentioned in section 3.2, the main shortcoming of the algorithm of [16] (from our perspective) is the running time. In particular, the running time of the τ-multiplicative approximation algorithm to LIS distance is $\widetilde{O}((\frac{w(\mathcal{B})}{\mathtt{loss}_f})^{O(1)})$, where $\mathtt{loss}_f$ is the number of points not on the LIS. We instead need a running time of $\widetilde{O}(\frac{w(\mathcal{B})}{\mathtt{loss}_f})$. As described in section 3.2, the τ-multiplicative approximation algorithm of [16] is iterative. We combine one level of this recursion with the additive δn approximation algorithm given by [16] to achieve the desired running time.

First, we sketch the argument showing that, if $\mathtt{loss}_f$ is small, then one level of recursion in the muliplicative approximation algorithm manages to classify many points. We argue the contrapositive, showing that if there are many points that cannot be classified by one level of recursion, then $\mathtt{loss}_f$ must be large.

In [16], the point characterization μ-safe is introduced. Essentially, a point that is μ-safe is comparable to all but a μ fraction of nearby points (For a formal statement of this definition, see [16]). It turns out that the μ-unsafe points roughly correspond to the points that cannot be classified by one level of recursion of the multiplicative algorithm. We are able to show that, if there are many μ-unsafe points, then $\mathtt{loss}_f$ must be large. We state this formally in the following lemma. (We say that an X-index is μ-safe if its match is μ-safe).

LEMMA 10.1. *For input $\mathcal{U}$, consider two μ-safe X-indices $i < j$ and let $I = (i, j)$, $\mathcal{B} = I \times Y(\mathcal{U})$. Suppose the number of μ-unsafe X-indices in I is at least $|I|/6$. Then $\frac{\mathtt{loss}_f}{w(\mathcal{B})} \geq \mu/12$.*

Due to space limitations, we omit the proof of this lemma.

From here, we observe that we can use this lemma to show that, after the first level of recursion of the multiplicative approximation algorithm of [16], the number of bad points that remain unclassified is roughly at least a μ-fraction of the total number of points that remain unclassified. From here, we focus only on the remaining unclassified points. Let m be the number of points that remain unclassified. We know by the previous observation that $\Omega(\mu m)$ of these points are bad, i.e. the number of such points contributing to $\mathtt{loss}_f$ is $\Omega(\mu m)$. Therefore, if we want a $(1+\mu)$ multiplicative approximation to $\mathtt{loss}_f$ when restricting to these unclassified points, we can afford $\Omega(\mu^2 m)$ additive error. As a result, it suffices to run the additive approximation algorithm of [16] with additive parameter $\Omega(\mu^2)$ on the restricted input. Putting these pieces together yields the desired result. We state this in the following theorem.

THEOREM 10.1. *There is an algorithm with running time $(1/\mu)^{O(1/\mu^2)}\frac{n}{\mathtt{loss}_f}\log^{c'} n$ that given any μ outputs a value $d \in [(1-\mu)\frac{\mathtt{loss}_f}{n}, (1+\mu)\frac{\mathtt{loss}_f}{n}]$*

Due to space limitations, we again omit the proof of this theorem.

From here what remains is to carry out the modifications that turn this algorithm into an algorithm for ulam distance, as described in section 6. We again leave out the remaining details due to space limitations.

References

[1] A. Abboud and A. Backurs and V. V. Williams, *Quadratic-Time Hardness of LCS and other Sequence Similarity Measures*, CoRR, abs/1501.07053 (2015). 1

[2] N. Ailon and B. Chazelle and S. Comandur and D. Liu, *Estimating the Distance to a Monotone Function*, Random Structures and Algorithms, 31 (2007), pp. 371–383. 3

[3] D. Aldous and P. Diaconis, *Longest increasing subsequences: from patience sorting to the Baik-Deift-Johannson theorem*, Bulletin of the American Mathematical Society, 36 (1999), pp. 413–432. 1

[4] A. Andoni and P. Indyk and R. Krauthgamer, *Overcoming the ℓ_1 non-embeddability barrier: algorithms for product matrices*, Proceedings of the 20th Symposium

on Discrete Algorithms (SODA), (2009), pp. 865–874. 1

[5] A. Andoni and H. L. Nguyen, *Near-Optimal Sublinear Time Algorithms for Ulam Distance*, Proceedings of the 21st Symposium on Discrete Algorithms (SODA), (2010), pp. 76–86. 1, 3, 6, 8, 12, 19

[6] Tugkan Batu and Funda Ergun and Joe Kilian and Avner Magen and Sofya Raskhodnikova and Ronitt Rubinfeld and Rahul Sami, *A sublinear algorithm for weakly approximating edit distance*, Proceedings of Symposium of Theory of Computing (STOC), (2003), pp. 316–324. 1

[7] A. Backurs and P. Indyk, *Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false)*,Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC), (2015), pp. 51–58. 1

[8] K. Bringmann and M. Kunnemann, *Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping*, Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS), (2015), pp. 79–97. 1

[9] M. Charikar and R. Krauthgamer, *Embedding the Ulam metric in ℓ_1*, Theory of Computing, 2 (2006), pp. 207–224. 1

[10] D. Critchlow, *Ulam's Metric*, Encyclopedia of Statistical Sciences, 9 (1988), pp. 379–380. 1

[11] F. Ergun and S. Kannan and R. Kumar and R. Rubinfeld and M. Viswanathan, *Spot-checkers*, Journal of Computer Systems and Sciences (JCSS), 60 (2000), pp. 717–751. 3

[12] J. W. Hunt and T. G. Szymanski, *A fast algorithm for computing longest common subsequences*, Commun. ACM, 20 (1977), pp. 350–353.

[13] Mitzenmacher, Michael and Upfal, Eli, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, 2005. 2

[14] Motwani, Rajeev and Raghavan, Prabhakar, *Randomized Algorithms*, Cambridge University Press, 1995. 2

[15] M. Parnas and D. Ron and R. Rubinfeld, *Tolerant Property Testing and Distance Approximation*, Journal of Computer and System Sciences, 6 (2006), pp. 1012–1042. 3

[16] M. Saks and C. Seshadhri, *Estimating the longest increasing sequence in polylogarithmic time*, Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), (2010), pp. 458–467. 1, 3, 5, 20