THEORY OF COMPUTING, Volume 15 (2), 2019, pp. 1–31
www.theoryofcomputing.org

Time Bounds for Streaming Problems

Raphaël Clifford* Markus Jalsenius† Benjamin Sach

Received March 13, 2017; Revised January 11, 2018; Published September 7, 2019

Abstract: We give tight cell-probe bounds for the time to compute convolution, multiplication and Hamming distance in a stream. The cell probe model is a particularly strong computational model and subsumes, for example, the popular word RAM model.

- We first consider online convolution where the task is to compute the inner product between a fixed n-dimensional vector and a vector of the n most recent values from a stream. One symbol of the stream arrives at a time and then each output symbol must be computed before the next input symbol arrives.
- Next we show bounds for online multiplication of two n-digit numbers where one of the multiplicands is known in advance and the other arrives one digit at a time, starting from the lower-order end. When a digit arrives, the task is to compute a single new digit from the product before the next digit arrives.
- Finally we look at the online Hamming distance problem where the Hamming distance is computed instead of the inner product.

For each of these three problems, we give a lower bound of $\Omega((\delta/w)\log n)$ time on average per output symbol, where δ is the number of bits needed to represent an input symbol and w is the cell or word size. We argue that these bounds are in fact tight within the cell probe model.

ACM Classification: F.2.2

AMS Classification: 68Q17

Key words and phrases: lower bounds, cell probe, streaming algorithms, online algorithms

Preliminary versions of these results first appeared in the Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP), 2011 [7] and the Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2013 [8].

*Supported by EPSRC Fellowship EP/J019283/1.
†Supported by EPSRC Fellowship EP/J019283/1.

 DOI: 10.4086/toc.2019.v015a002

1 Introduction

We consider the complexity of three related and fundamental problems: computing the convolution of two vectors, multiplying two integers, and computing the Hamming distance between two strings. We study these problems in an online or streaming context and provide matching upper and lower bounds in the cell-probe model. Time lower bounds in the cell-probe model also hold for the popular word-RAM model in which many of today's algorithms are given.

The importance of these problems is hard to overstate. The integer multiplication and convolution problems have played a central role in modern algorithm design and theory. The question of how to compute the Hamming distance efficiently has a rich literature, spanning many of the most important fields in computer science. Within the theory community, communication complexity based lower bounds and streaming model upper bounds for the Hamming distance problem have been the subject of particularly intense study [10, 35, 17, 19, 4, 5]. This previous work has however almost exclusively focused on providing resource bounds either in terms of space or bits of communication rather than time complexity.

We begin by introducing the problems and stating our results. In the following problem definitions and throughout, we write $[q]$ to denote the set $\{0,\ldots,q-1\}$, where q is a positive integer and a parameter of the problem.

Problem 1.1 (Online convolution). For a fixed vector $F \in [q]^n$ of length n, we consider a stream in which numbers from $[q]$ arrive one at a time. For each arriving number, before the next number arrives, we compute the inner product (modulo q) of F and the vector that consists of the most recent n numbers of the stream.

Theorem 1.2. *In the cell-probe model with w bits per cell, for any positive integers q and n, and any randomised algorithm solving the online convolution problem, there exist instances such that the expected amortised time is $\Omega((\delta/w)\log n)$ per arriving value, where $\delta = \lceil \log_2 q \rceil$.*

Problem 1.3 (Online multiplication). Given two numbers $F, X \in [q^n]$, where q is the base and n is the number of digits per number, we want to compute the n least significant digits of the product of F and X, in base q. We must do this under the constraint that only F is known in advance and the digits of X arrive one at a time, starting from the lower-order end. When the i-th digit of X arrives, before the $(i+1)$-th digit arrives, we compute the i-th digit of the product.

Theorem 1.4. *In the cell-probe model with w bits per cell, for any positive integers q and n, and any randomised algorithm solving the online multiplication problem in base q, there exist instances such that computing the n least significant digits of the product takes $\Omega((\delta/w)n\log n)$ expected time, where $\delta = \lceil \log_2 q \rceil$.*

Problem 1.5 (Online Hamming distance). For a fixed string F of length n, we consider a stream in which symbols from the alphabet $[q]$ arrive one at a time. For each arriving symbol, before the next symbol arrives, we compute the Hamming distance between F and the last n symbols of the stream.

Theorem 1.6. *In the cell-probe model with w bits per cell, for any positive integers q and n, and any randomised algorithm solving the online Hamming distance problem, there exist instances such that the expected amortised time is $\Omega((\delta/w)\log n)$ per arriving value, where $\delta = \lceil \min\{\log_2 q, \log_2 n\} \rceil$.*

Our Hamming distance lower bound also implies a matching lower bound for any problem to which Hamming distance can be reduced. The most straightforward of these is online L_1 distance computation, where the task is to compute the L_1 distance between a fixed vector of integers and the last n numbers in the stream. A suitable reduction was shown in [26]. The expected amortised cell probe complexity for the online L_1 distance problem is therefore also $\Omega((\delta/w)\log n)$ per output symbol.

One of our main technical contributions is to extend methods designed to give lower bounds on dynamic data structures to the seemingly distinct field of online algorithms. Where $\delta = w$, for example, we have $\Omega(\log n)$ lower bounds for all three problems. In particular for online multiplication and convolution, these lower bounds match the best currently known offline upper bounds in the RAM model. As we discuss in Section 1.1, this may be the highest lower bound that can be proved for all the problems we consider without a further breakthrough.

In order to prove our lower bounds we show the existence of probability distributions on the inputs for which we can prove lower bounds on the expected running time of any deterministic algorithm. By Yao's minimax principle [36] this immediately implies that for every (randomised) algorithm there is a worst-case input such that the (expected) running time is equally high. Therefore our lower bounds hold equally for randomised algorithms as for deterministic ones.

The lower bounds we give are also tight within the cell-probe model. This can be seen by application of reductions described in [12, 6]. It was shown there that any offline algorithm for convolution [6] or multiplication [12] can be converted to an online one with at most an $O(\log n)$ factor overhead. For details of these reductions we refer the reader to the original papers. In our case, the same approach also allows us to directly convert any cell-probe algorithm from an offline to online setting. An offline cell-probe algorithm for convolution, multiplication or Hamming distance could first read the whole input, then compute the answer. This takes $O((\delta/w)n)$ cell probes. We can therefore derive online cell-probe algorithms which take only $O((\delta/w)n\log n)$ probes over n input symbols, hence $O((\delta/w)\log n)$ (amortised) probes per output. This upper bound matches the new lower bounds we give. We summarise this in the following corollary.

Corollary 1.7. *The expected amortised cell-probe complexity of the online convolution, multiplication, Hamming distance and L_1-distance problems is $\Theta((\delta/w)\log n)$ per arriving value.*

One consequence of our results is the first strict separation between the complexity of exact and approximate pattern matching. Online exact matching can be solved in constant time [15] per new input symbol and our new lower bound proves for the first time that this is not possible for Hamming distance.

Another consequence of our results is a new separation between the time complexity of online exact matching and any convolution-based online pattern matching algorithm. Convolution has played a particularly important role in the field of combinatorial pattern matching where many of the fastest algorithms rely crucially for their speed on the use of fast Fourier transforms (FFTs) to perform repeated convolutions. These methods have also been extended to allow searching for patterns in rapidly processed data streams [6, 9].

1.1 Previous results and upper bounds in the RAM model

Almost all previous algorithmic work for exact Hamming distance computation has considered the problem in an offline setting. Given a pattern P and a text T of length m and n respectively, the best current deterministic upper bound for offline Hamming distance computation is an $O(n\sqrt{m\log m})$ time algorithm based on convolutions [2, 22]. In [21] a randomised algorithm was given that takes $O((n/\varepsilon^2)\log^2 n)$ time which was subsequently modified in [18] to $O((n/\varepsilon^3)\log n)$. Particular interest has also been paid to a bounded version of this problem called the k-mismatch problem. Here a bound k is given and we need only report the Hamming distance if it is less than or equal to k. In [23], an $O(nk)$ algorithm was given that is not convolution based and uses $O(1)$-time lowest common ancestor (LCA) operations on the suffix tree of P and T. This was then improved to $O(n\sqrt{k\log k})$ time by a method that combines LCA queries, filtering and convolutions [3].

The best time complexity lower bounds for online multiplication of two n-bit numbers were given in the 1974 by Paterson, Fischer and Meyer. They presented an $\Omega(\log n)$ lower bound for multitape Turing machines [31] and also gave an $\Omega(\log n/\log\log n)$ lower bound for the *bounded activity machine* (BAM). The BAM, which is a strict generalisation of the Turing machine model but which has nonetheless largely fallen out of favour, attempts to capture the idea that future states can only depend on a limited part of the current configuration. To the authors' knowledge, there has been no progress on cell-probe lower bounds for online multiplication, convolution or Hamming distance previous to the work we present here.

There have however been attempts to provide offline lower bounds for the related problem of computing the FFT. In [28] Morgenstern gave an $\Omega(n\log n)$ lower bound conditional on the assumption that the underlying field of the transform is the complex numbers and that the modulus of any complex numbers involved in the computation is at most one. Papadimitriou gave the same $\Omega(n\log n)$ lower bound for FFTs of length a power of two, this time excluding certain classes of algorithms including those that rely on linear mathematical relations among the roots of unity [30]. This work had the advantage of giving a conditional lower bound for FFTs over more general algebras than was previously possible, including for example finite fields. In 1986, Pan [29] showed that another class of algorithms having a so-called synchronous structure must require $\Omega(n\log n)$ time for the computation of both the FFT and convolution.

The fastest known algorithms for both offline integer multiplication and convolution in the word-RAM model require $O(n\log n)$ time by a well known application of a constant number of FFTs. As a consequence our online lower bounds for these two problems match the best known time upper bounds for the offline problem. As we discussed above, our lower bounds for all three problems are also tight within the cell-probe model for the online problems.

The question now naturally arises as to whether one can find higher lower bounds in the RAM model. This appears as an interesting question as there remains a gap between the best known time upper bounds provided by existing algorithms and the lower bounds that we give within the cell-probe model. However, as we mention above, any offline algorithm for convolution, Hamming distance or multiplication can be converted to an online one with at most an $O(\log n)$ factor overhead [12, 6]. As a consequence, a higher lower bound than $\Omega(\log n)$ for any of these problems would immediately imply a superlinear lower bound for the offline version of the corresponding problem. This would be a truly remarkable breakthrough in the field of computational complexity as no such offline lower bound is known even for the canonical NP-complete problem SAT.

Our only alternative route to find tight time bounds would be to find better upper bounds for the online problems. For the case of online multiplication at least, where the fastest online RAM algorithm takes $O(\log^2 n)$ time per arriving pair of digits, this has been an open problem since at least 1973 and has so far resisted our best attempts. On the other hand, for online Hamming distance, while our lower bound is tight within the model, it is still distant from the time complexity of the fastest known RAM algorithms. The best known online complexity is $O(\sqrt{n \log n})$ time per arriving symbol [6]. An improvement of the upper bound for Hamming distance computation to meet our new lower bound would also have significant implications. A reduction that is now regarded as folklore tells us that any $O(f(n))$ time algorithm for computing the Hamming distance between a pattern and all substrings of a text, assuming a pattern of length n and a text of length $2n$, implies an $O(f(n^2))$ time algorithm for multiplying binary $(n \times n)$-matrices over the integers. Therefore an $O(\log n)$-time online Hamming distance algorithm would imply an $O(n \log n)$ offline Hamming distance algorithm, which would in turn imply an $O(n^2 \log n)$-time algorithm for binary matrix multiplication. Although such a result would arguably be less shocking than a proof of a superlinear offline lower bound for Hamming distance computation, it would nonetheless be a significant breakthrough in the complexity of a classic and much studied problem.

1.2 The cell-probe model

Our bounds hold in the *cell-probe model* which is a particularly strong computational model that was introduced originally by Minsky and Papert [27] in a different context and then subsequently by Fredman [13] and Yao [37]. A data structure in the cell-probe model consists of a set of memory cells, each storing w bits. When presented with an update, the data structure reads and updates a number of these cells. Similarly, when presented with a query, the data structure reads cells and from their contents, returns the desired answer. These reads and updates are referred to as cell probes and the cost of an update or query operation is simply the number of cells that are probed. As is typical, we will require that the cell size w is at least of order $\log n$ bits. This allows each cell, or a constant number of cells, to hold the address of any location in memory.

This abstraction makes the model very strong, subsuming for instance the popular word-RAM model. In the word-RAM model certain operations on words, such as addition, subtraction and possibly multiplication take constant time (see for example [16] for a detailed introduction). Here a word corresponds to a cell. Any time lower bound in the cell-probe model with w-bit cells gives an asymptotically equal time lower bound in the word-RAM model with w-bit words. This is because each constant-time operation in the word-RAM model only probes a constant number of cells.

The generality of the cell-probe model makes it particularly attractive for establishing lower bounds for dynamic data structure problems and many such results have been given in the past couple of decades. The approaches taken had historically been based only on communication complexity arguments and the chronogram technique of Fredman and Saks [14]. However in 2004, a breakthrough lead by Pătraşcu and Demaine gave us the tools to seal the gaps for several data structure problems [34] as well as giving the first $\Omega(\log n)$ lower bounds. The new technique is based on information-theoretic arguments that we also deploy here. Pătraşcu and Demaine also presented ideas which allowed them to express more refined lower bounds such as trade-offs between updates and queries of dynamic data structures. For a list of data structure problems and their lower bounds using these and related techniques, see for example [32]. A new lower bound of $\Omega\left((\log n/\log\log n)^2\right)$ was given by Larsen in 2012 for the cell-probe complexity of performing queries in the dynamic range counting problem [24]. This result holds under the natural assumptions of $\Theta(\log n)$-size words and polylogarithmic time updates and is another exciting breakthrough in the field of cell-probe complexity. In a recent significant advance for the field, an $\Omega((\log^{1/2} n/\log\log n)^3)$ time lower bound for the unweighted version of dynamic range counting was given which holds even over $\mathbb{F}_2$ [25].

1.3 Technical contributions

We use one of the most important techniques for proving data structure lower bounds called the *information transfer method* of Pătraşcu and Demaine [33, 34]. For a pair of adjacent intervals of arriving values in the stream, the information transfer is the set of memory cells that are written during the first interval and read in the next interval. These cells must contain *all* the information from the updates during the first interval that the algorithm needs in order to produce correct output in the next interval. If one can prove that this quantity is large for many pairs of intervals then the desired lower bound follows. To do this we relate the size of the information transfer to the conditional entropy of the output in the relevant interval. The main task of proving lower bounds reduces to that of devising a hard input distribution for which output symbols have high entropy conditioned on selected previous values of the input.

Although the use of information transfer to provide time lower bounds for data structure problems is not new, applying the method to our new online setting has required a number of new insights. At the simplest level, where a standard data structure problem has a number of different possible queries, in our setting there is only one query which is to return the latest result as soon as a new symbol arrives. As a result we provide a complete description of the information transfer method in a form which is relevant to this different setting. At a more detailed mathematical level, perhaps the most surprising innovation we present is a new relationship between the Hamming distance, vector sums and constant-weight binary cyclic codes.

For the three problems we consider, our key contribution is the design of a fixed vector or string F which together with some random distribution over possible input streams provide a lower bound for the information transfer between successive intervals. For the convolution and multiplication problems we show that a randomly picked F has a good chance of being suitable for proving the lower bounds. We also give an explicit description of a particular F for which the lower bounds are obtained when the values of the input stream are drawn independently and uniformly at random. The vector F is easy to describe and naturally yields large conditional entropy of the output symbols for intervals of power-of-two lengths.

The results of the convolution and multiplication problems can be seen as a first step towards the lower bound for the Hamming distance problem. Here the string F is derived by a sequence of transformations. These start with binary cyclic codes and go via binary vectors with many distinct sums and an intermediate string to finally arrive at F itself. The use of such a purposefully designed input departs from the closely related work of the convolution and multiplication lower bounds and also from much of the lower bound literature where simple uniform distributions over the whole input space often suffice.

The central fact that enabled a lower bound to be proven for the online convolution problem is that the inner product between a vector and successive suffixes of the stream reveals a lot of information about the history of the stream. Establishing a similar result for online Hamming distance problem appears, however, to be considerably more challenging for a number of reasons. The first and most obvious is that the amount of information one gains by comparing whether two, potentially large, symbols are equal is at most one bit, as opposed to $O(\log n)$ bits for multiplication. The second is that the particularly simple worst-case vector F of the convolution problem greatly eased the resulting analysis. We have not been able to find such a simple fixed string for the Hamming distance problem and our proof of the existence of a hard instance is non-constructive and involves a number of new insights, combining ideas from coding theory and additive combinatorics.

When computing the Hamming distance there is a balance between the number of symbols being used and the length of the strings. For large alphabets and short strings, one would expect a typical Hamming distance to be close to the length of the string on random input symbols and therefore to provide very little information about the random string itself. This suggests that the length of the strings must be sufficiently long in relation to the alphabet size to ensure that the entropy of the output symbols is large, as required by the information transfer method. At first glance, it is not immediately obvious that large entropy can be obtained unless the fixed string F is *exponentially* larger than the alphabet size. This potentially poses another problem for the information transfer method, namely that a word size w of order $\log n$ would be much larger than δ (the number of bits needed to represent a symbol), making a $\log n$ lower bound impossible to achieve.

Our main technical contribution is to show that fixed strings of length only polynomial in the size of the alphabet exist which provide output symbols of sufficiently high entropy. Such strings, when combined with a suitable input distribution maximising the number of distinct Hamming distance output sequences, give us the overall lower bound. We design a fixed string F with this desirable property in such a way that there is a one-to-one mapping between many of the different possible input streams and the computed Hamming distances. This in turn implies large entropy. The construction of F is non-trivial and we break it into smaller building blocks, reducing our problem to a purely combinatorial question relating to vectors sums. That is, given a relatively small set V of vectors of length m, how many distinct vector sums can be obtained by choosing m vectors from V and adding them. We show that even if we are restricted to picking vectors only from subsets of V, there exists a V such that the number of distinct vector sums is $m^{\Omega(m)}$. We believe this result is interesting in its own right. Our proof for the combinatorial problem is non-constructive and probabilistic, using constant-weight cyclic binary codes to prove that there is a positive probability of the existence of a set V with the desired property.

1.4 Organisation

In Section 2 we introduce notation and describe the setup for proving the lower bounds. In Section 3 we prove the lower bounds for all three problems that we consider. The proofs hinge on a set of lemmas which give lower bounds related to the entropy of the outputs of the problems considered. These lemmas are proven separately in subsequent sections. In Section 4 we deal with the lemmas related to the convolution problem, and in Section 5 we deal with the lemmas related to the multiplication problem. Finally, in Sections 6, 7 and 8 we prove the lemma related to the Hamming distance problem.

2 Basic setup for the lower bounds

In this section we introduce notation and concepts that are used heavily in the lower bound proofs. For an array, vector or string A of length n and $i, j \in [n]$, we write $A[i]$ to denote the value at position i, and where $j \geq i$, $A[i, j]$ denotes the length-$(j-i+1)$ subarray of A starting at position i. All logarithms are in base two. We first introduce a unifying framework for the problems we consider.

2.1 The framework

There is a *fixed array* F and an array S which is referred to as the *stream*. Both F and S are of length n and over the set $[q]$ of integers, and we let $\delta = \lceil \log q \rceil$ denote the number of bits required to encode a value from $[q]$. The value q, or alternatively δ, is a parameter of the problem. The problem is to maintain S subject to an update operation $\text{UPDATE}(x)$ which takes a symbol $x \in [q]$, modifies S by appending x to the right of the rightmost symbol $S[n-1]$ and removing the leftmost symbol $S[0]$, and then outputs the value of a function of F and the updated S. In the *convolution* problem the output is the inner product of F and S, that is $\sum_{i\in[n]}(F[i] \cdot S[i])$, and in the *Hamming distance* problem the output is the number of positions $i \in [n]$ such that $F[i] \neq S[i]$.

We let $U \in [q]^n$ denote the *update array* which describes a sequence of n UPDATE operations. That is, for each $t \in [n]$, the operation $\text{UPDATE}(U[t])$ is performed. We will usually refer to t as the *arrival* of the value $U[t]$. Observe that just after the arrival t, the values $U[t+1, n-1]$ are still not known to the algorithm. Finally, we let the length-n array A denote the outputs such that for $t \in [n]$, $A[t]$ is the output of $\text{UPDATE}(U[t])$.

In the *multiplication* problem we let F denote one of the two operands to be multiplied, hence F is fixed and known in advance by the algorithm. Specifically we let $F[i]$ denote the i-th least significant digit. We let U be the unknown operand so that $U[t]$ is its t-th least significant digit. Prior to the arrival of the first digit $U[0]$, the stream S contains only zeros. The output $A[t]$ is the t-th digit in the product of F and S, which is a function of F and $U[0,t]$ as required.

2.2 Hard distributions

Our lower bounds hold for any randomised algorithm on its worst case input. This will be achieved by applying *Yao's minimax principle* [36]. That is, we develop lower bounds that hold for any deterministic algorithm on some random input. The basic approach is as follows: we devise a fixed array F and describe a probability distribution for n new values arriving in the stream S. We then obtain a lower bound on the expected running time for any deterministic algorithm over these arrivals. Due to the minimax principle, the same lower bound must then hold for any randomised algorithm on its own worst case input. The amortised bound is obtained by dividing by n.

From this point onwards we consider an arbitrary deterministic algorithm running with some fixed array F on a random input of n values. The algorithm may depend on F. We refer to the choice of F and distribution on U as a *hard distribution* since it is used to show a lower bound.

2.3 Information transfer

The *information transfer tree*, denoted $\mathcal{T}$, is a balanced binary tree over n leaves. To avoid technicalities we assume that n is a power of two. The leaves of $\mathcal{T}$, from left to right, represent the arrivals t from 0 to $n-1$. For a node v of $\mathcal{T}$, we let ℓ_v denote the number of leaves in the subtree rooted at v. An internal node v is associated with three arrivals, t_0, t_1 and t_2. Here t_0 is the arrival represented by the leftmost node in subtree rooted at v, similarly $t_2 = t_0 + \ell_v - 1$ is the rightmost such node and $t_1 = t_0 + \ell_v/2 - 1$ is in the middle. That is, the intervals $[t_0, t_1]$ and $[t_1 + 1, t_2]$ span the left and right subtrees of v, respectively. For example, in Figure 1, the node labelled v is associated with the intervals $[16, 23]$ and $[24, 31]$.

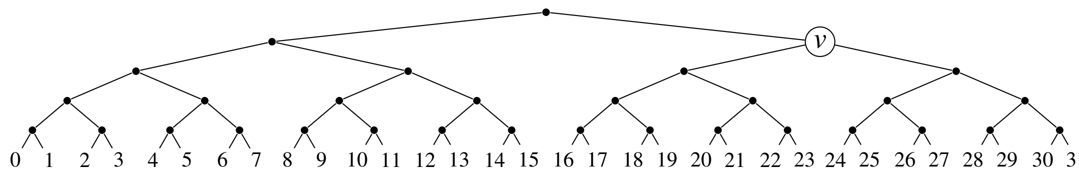

Figure 1: An information transfer tree $\mathcal{T}$ with $n = 32$ leaves. For the node labelled v, the arrival times $t_0 = 16$, $t_1 = 23$ and $t_2 = 31$.

We define the subarray $U_v = U[t_0, t_1]$ to represent the $\ell_v/2$ values arriving in the stream during the arrival interval $[t_0, t_1]$, and we define the subarray $A_v = A[t_1 + 1, t_2]$ to represent the $\ell_v/2$ output symbols produced during the arrival interval $[t_1 + 1, t_2]$.

We define $\widetilde{U}_v$ to be the concatenation of $U[0, (t_0 - 1)]$ and $U[(t_2 + 1), (n - 1)]$. That is, $\widetilde{U}_v$ contains all symbols of U except for those in U_v. When $\widetilde{U}_v$ is fixed to some constant $\widetilde{u}_v$ and U_v is random, we write $H(A_v \mid \widetilde{U}_v = \widetilde{u}_v)$ to denote the conditional entropy of A_v under the fixed $\widetilde{U}_v$.

We define the *information transfer* of a node v of $\mathcal{T}$, denoted $\mathcal{I}_v$, to be the set of memory cells c such that c is probed during the interval $[t_0, t_1]$ and also probed in $[t_1 + 1, t_2]$. The cells in the information transfer $\mathcal{I}_v$ therefore contains all the information about the values in U_v that the algorithm uses in order to correctly produce the output symbols A_v.

By adding up the sizes of the information transfers $\mathcal{I}_v$ over the internal nodes v of $\mathcal{T}$ we get a lower bound on the number of cell probes, that is a lower bound on the total running time of the algorithm. To see this it is important to make the observation that a particular cell probe is counted for only once. Suppose that the cell $c \in \mathcal{I}_v$ for some node v. Let p be the first probe of c in the arrival interval $[t_1 + 1, t_2]$. By including the cell $c \in \mathcal{I}_v$ in the cell probe count we are in fact counting the probe p. Now observe that p cannot be counted for in the information transfer $\mathcal{I}_{v'}$ of any node v' where v' is a proper descendant or ancestor of v.

Since the concept of the size of the information transfer is central to the lower bound proofs, we define as a shorthand $I_v = |\mathcal{I}_v|$ to denote the size of the information transfer.

Definition 2.1 (Large expected information transfer). A node v of $\mathcal{T}$ has *large information transfer* if

$$\mathbb{E}[I_v] \geq \frac{k \cdot \delta \cdot \ell_v}{w},$$

where k is a constant that depends on the problem and input distribution.

Our aim is to show that a substantial proportion of the nodes of $\mathcal{T}$ have large information transfer. As we will see in Section 3, this will be achieved by relating the size of the information transfer, I_v to the entropy of the output symbols A_v.

3 Overall proofs of the lower bounds

In this section we give the overall proofs for our lower bound results. Consider any node v in $\mathcal{T}$. Suppose that $\widetilde{U}_v$ is fixed arbitrarily but the symbols in U_v are randomly drawn in accordance with the distribution on U, conditioned on the fixed value of $\widetilde{U}_v$. This induces a distribution on the output symbols A_v. If the entropy of A_v is large, conditioned on the fixed $\widetilde{U}_v$, then any algorithm must probe many cells in order to correctly produce the output symbols A_v, as it is only through the information transfer $\mathcal{I}_v$ that the algorithm can know anything about U_v. We will begin in Section 3.1 by making this claim precise by giving a problem independent upper bound on $H(A_v \mid \widetilde{U}_v = \widetilde{u}_v)$ in terms of $\mathcal{I}_v$. We will then give lower bounds on $H(A_v \mid \widetilde{U}_v = \widetilde{u}_v)$ in Section 3.2 for each problem we consider. The proofs of these entropy lower bounds form the heart of our contributions and are deferred to Sections 4 onwards. In Section 3.3, we combine the upper and lower bounds on the entropy to show that many nodes of $\mathcal{T}$ have large information transfer. Finally, we calculate our final lower bounds in Section 3.4 by summing over all large information transfer nodes as discussed in Section 2.3 above.

3.1 An upper bound on the entropy

Towards showing that high conditional entropy $H(A_v \mid \widetilde{U}_v = \widetilde{u}_v)$ implies large information transfer we use the information transfer $\mathcal{I}_v$ to describe an encoding of the output symbols A_v. The following lemma gives a direct relationship between the size of the information transfer $\mathcal{I}_v$ and the entropy. The lemma was originally stated in [34] but for completeness we restate it here in our notation and provide a full proof.

Lemma 3.1 (Pătraşcu and Demaine [34]). *Under the assumption that the address of any cell can be specified in w bits, for any node v of the information transfer tree $\mathcal{T}$, the entropy*

$$H(A_v \mid \widetilde{U}_v = \widetilde{u}_v) \;\leq\; w + 2w \cdot \mathbb{E}[I_v \mid \widetilde{U}_v = \widetilde{u}_v]\,.$$

Proof. The expected length of any encoding of A_v, conditioned on $\widetilde{U}_v$, is an upper bound on the conditional entropy of A_v. We use the information transfer $\mathcal{I}_v$ as an encoding in the following way. For every cell $c \in \mathcal{I}_v$ we store the address of c, which takes at most w bits under the assumption that a cell can hold the address of any cell in memory. We also store the contents of c, which takes w bits. In total this requires $2w \cdot I_v$ bits. We will use the algorithm, which is fixed, and the fixed values of $\widetilde{U}_v$ as part of the decoder to obtain A_v from the encoding. Since the encoding is of variable length we also store the size of the information transfer, which requires at most w additional bits.

In order to prove that the described encoding of A_v is valid we now describe how to decode it. First we simulate the algorithm on the fixed input $\widetilde{U}_v$ from the first arrival of $U[0]$ until just before the first value in U_v arrives. We then skip over all input symbols in U_v and resume simulating the algorithm from the beginning of the interval where A_v is computed until the last value in A_v has been obtained. For every cell being read, we check if it is contained in information transfer $\mathcal{I}_v$ by looking up its address in the encoding. If it is in the information transfer, its contents is fetched from the encoding. If not, its contents is available from simulating the algorithm on the fixed input symbols. Observe that it suffices to store only the first time a cell in the information transfer is probed as the decoder remembers every cell it has already accessed. □

3.2 Lower bounds on the entropy

Lemma 3.1 above provides a direct way to obtain a lower bound on the expected size of the information transfer given a lower bound on the conditional entropy $H(A_v \mid \widetilde{U}_v = \widetilde{u}_v)$. To show that a node has large information transfer we introduce the following definition.

Definition 3.2 (High-entropy node). A node v in $\mathcal{T}$ is a *high-entropy node* if there is a positive constant k such that for *any* fixed $\widetilde{u}_v$,

$$H(A_v \mid \widetilde{U}_v = \widetilde{u}_v) \geq k \cdot \delta \cdot \ell_v \,.$$

To put this bound in perspective, note that the maximum conditional entropy of A_v is bounded by the entropy of U_v, which is at most $\delta \cdot (\ell_v/2)$ and obtained when the values of U_v are independent and uniformly drawn from $[q]$. Thus, the conditional entropy associated with a high-entropy node is the highest possible up to some constant factor. Establishing high-entropy nodes is the main contribution of this paper and the results are given in the following lemmas.

Lemma 3.3. *For the convolution problem, suppose that U is chosen uniformly at random from $[q]^n$, where q is a prime. For any $v \in \mathcal{T}$, at least a $(1-1/q)$-fraction of all $F \in [q]^n$ have the property that v is a high-entropy node.*

The proof of the above lemma is given in Section 4 and relies on properties of Toeplitz matrices over a finite field of q elements. The proof does not give explicit descriptions of fixed arrays F for which nodes are high-entropy nodes. In the proof of the next lemma however, we show that there exists a particular array F for which high-entropy nodes are obtained. This F is a 0/1-array and is easy to describe: zeroes everywhere except for at power-of-two positions from the right hand end. The proof is given in Section 4.

Lemma 3.4. *For the convolution problem there exists a fixed array $F \in [q]^n$ such that when U is chosen uniformly at random from $[q]^n$, all $v \in \mathcal{T}$ are high-entropy nodes.*

Before we give the lemmas concerning online multiplication, recall that in this problem there is a fixed operand F multiplied with an operand U for which digits arrive one at a time.

Lemma 3.5. *For the online multiplication problem, suppose that the operand U is chosen uniformly at random from $[q^n]$. For any $v \in \mathcal{T}$, at least half of all operands $F \in [q^n]$ have the property that v is a high-entropy node.*

The proof of Lemma 3.5 is given in Section 5. Similarly to the convolution problem we also give an explicit description of a number F for which high-entropy nodes are obtained. This number resembles the fixed array that we described above for the convolution problem. The proof of the next lemma is also given in Section 5.

Lemma 3.6. *For the online multiplication problem there exists a fixed operand $F \in [q^n]$ such that when U is chosen uniformly at random from $[q^n]$, all $v \in \mathcal{T}$ are high-entropy nodes.*

Finally, for the Hamming distance problem we show that there exists an F and distribution for U such that sufficiently many nodes are high-entropy nodes. The proof of the next lemma is rather involved and is given over Sections 6 to 8.

Lemma 3.7. *For the Hamming distance problem there exists a hard distribution with a fixed F and random U such that any node $v \in \mathcal{T}$ for which $\ell_v \geq h \cdot \sqrt{n}$ is a high-entropy node, where h is a positive constant.*

In the proof of Lemma 3.7 we demonstrate that there exists a very specific set of strings such that when F is drawn randomly from this set, there is a non-zero probability of picking an F for which many nodes are high-entropy nodes. Unlike the convolution and multiplication problems, the distribution for U is not uniform over of all strings $[q]^n$.

3.3 Lower bounds on the information transfer

In the previous section we gave a series of lemmas saying that for all three problems we consider, there are instances for which many nodes of $\mathcal{T}$ are high-entropy nodes. In this section we combine these results with the entropy upper bound of Lemma 3.1 to show that many nodes have large information transfer. The following lemmas match the lemmas of the previous section. We start with the convolution problem.

Lemma 3.8. *For the convolution problem where both F and U are chosen uniformly at random from $[q]^n$, and q is a prime, every $v \in \mathcal{T}$ has large information transfer.*

Proof. By combining Lemmas 3.1 and 3.3 we have that for any $v \in \mathcal{T}$ under fixed $\widetilde{U}_v$, at least half of all $F \in [q]^n$ imply that v is a high-entropy node, that is,

$$k \cdot \delta \cdot \ell_v \;\leq\; w + 2w \cdot \mathbb{E}[I_v \mid \widetilde{U}_v = \widetilde{u}_v]\,,$$

where k is the constant from Definition 3.2 of a high-entropy node. Rearranging terms gives

$$\mathbb{E}[I_v \mid \widetilde{U}_v = \widetilde{u}_v] \;\geq\; \frac{\delta \cdot \ell_v}{2k \cdot w} - \frac{1}{2}\,.$$

We remove the conditioning by taking expectation over $\widetilde{U}_v$ under a random U. When F is chosen uniformly at random from $[q]^n$ we therefore have

$$\mathbb{E}[I_v] \;\geq\; \frac{\delta \cdot \ell_v}{4k \cdot w} - \frac{1}{4}\,,$$

hence v has large information transfer. □

Similarly to Lemma 3.8, we combine Lemmas 3.1 and 3.4 to obtain the following property for the case where F is a fixed string and not randomly chosen.

Lemma 3.9. *For the convolution problem there exists a hard distribution where F is fixed and U is chosen uniformly at random from $[q]^n$, such that every $v \in \mathcal{T}$ has large information transfer.*

Proof. Similarly to the proof of Lemma 3.8 we combine Lemmas 3.1 and 3.4 to obtain, for all $v \in \mathcal{T}$ under fixed $\widetilde{U}_v$,

$$\mathbb{E}[I_v \mid \widetilde{U}_v = \widetilde{u}_v] \;\geq\; \frac{\delta \cdot \ell_v}{2k \cdot w} - \frac{1}{2}\,,$$

where k is the constant from Definition 3.2 of a high-entropy node. The conditioning is removed by taking expectation over $\widetilde{U}_v$ under a random U. □

The proofs of the following two lemmas, in which we establish large information transfer for the multiplication problem, are similar to the proofs of the previous two lemmas, only that we here combine Lemma 3.1 with Lemmas 3.5 and 3.6, respectively.

Lemma 3.10. *For the online multiplication problem where both operands are chosen uniformly at random from $[q^n]$, every $v \in \mathcal{T}$ has large information transfer.*

Lemma 3.11. *For the online multiplication problem there exists a fixed operand in $[q^n]$ such that when the other operand is chosen uniformly at random from $[q^n]$, every $v \in \mathcal{T}$ has large information transfer.*

Finally, large information transfer is also established for the Hamming distance problem. The proof of the next lemma is identical to the proof of Lemma 3.9, only that we combine Lemma 3.1 with Lemma 3.7 instead, and restrict the nodes v to those for which ℓ_v is greater than a constant times $\sqrt{n}$.

Lemma 3.12. *There exists a hard distribution for the Hamming distance problem such that every $v \in \mathcal{T}$ for which $\ell_v \geq h \cdot \sqrt{n}$ has large information transfer, where h is a positive constant.*

3.4 Obtaining the cell-probe lower bounds

Now that we have established large information transfer for sufficiently many nodes of $\mathcal{T}$ we are ready to prove the lower bounds of Theorems 1.2, 1.4 and 1.6.

For both the convolution and multiplication problems, large information transfer has been established for every node v of $\mathcal{T}$, whereas for the Hamming distance problem, large information transfer has only been established where $\ell_v \geq h \cdot \sqrt{n}$ where h is a positive constant. In order to unify the presentation of the proofs we restrict the summation of I_v to nodes for which $\ell_v \geq h \cdot \sqrt{n}$. Let V denote this set of nodes. We have

$$\mathbb{E}\left[\sum_{v \in \mathcal{T}} I_v\right] \geq \mathbb{E}\left[\sum_{v \in V} I_v\right] = \sum_{v \in V} \mathbb{E}[I_v] \geq \sum_{v \in V} \frac{k \cdot \delta \cdot \ell_v}{w} = \frac{k' \cdot \delta \cdot n \cdot \log n}{w}, \tag{3.1}$$

where k is the constant from Definition 2.1 of large information transfer and k' is a new suitable constant. The first equality follows by linearity of expectation and the second inequality follows by Lemmas 3.8 to 3.12, respectively. The last equality follows from the fact that

$$\sum_{\substack{v \in T \\ \ell_v \geqslant h \cdot \sqrt{n}}} \ell_v \in \Theta(n \log n).$$

Since the running time is bounded by the number of cell probes we have from Equation (3.1) that the expected running time for any deterministic algorithm solving the convolution, multiplication or Hamming distance problem, respectively, on n random input symbols is

$$\Omega\left(\frac{\delta \cdot n \cdot \log n}{w}\right).$$

By Yao's minimax principle, as discussed in Section 2, this implies that any randomised algorithm on its worst case input has the same lower bound on its expected running time. The amortised time per arriving value is obtained by dividing the running time by n. This concludes the proofs of Theorems 1.2, 1.4 and 1.6.

4 Hard distributions for the convolution problem

In this section we prove Lemmas 3.3 and 3.4, that is we show that there are instances to the convolution problem such that the conditional entropy of the output symbols A_v is large, where all input symbols but U_v are fixed.

We begin by proving Lemma 3.3 because the proof is straightforward and the description of the hard distribution is simple: pick the input symbols U uniformly at random from $[q]^n$. As to the choice of F we only argue that a large fraction of all length-n arrays have the desired entropy lower bound. In Section 4.2 we will specify a particular F with this property, which will lead to a proof of Lemma 3.4.

4.1 Entropy lower bound over all arrays F

We now prove Lemma 3.3. Let v be any internal node of $\mathcal{T}$ and let $t_v \in [n]$ denote the arrival time of $U_v[0]$. Let $\ell = \ell_v/2$. For $i \in [\ell]$, the i-th output in A_v can be broken into two sums $\mathcal{A}_i$ and $\widetilde{\mathcal{A}}_i$, such that $A_v[i] = \mathcal{A}_i + \widetilde{\mathcal{A}}_i$, where

$$\mathcal{A}_i = \sum_{j \in [\ell]} \big(F[n-1-(\ell+i)+j] \cdot U_v[j]\big)$$

is the contribution from the alignment of F with U_v, and $\widetilde{\mathcal{A}}_i$ is the contribution from the alignments that do not include U_v. Hence $\widetilde{\mathcal{A}}_i$ is constant under fixed $\widetilde{U}_v$. We define $M_{F,\ell}$ to be the $\ell \times \ell$ matrix with entries $M_{F,\ell}(i,j) = F[n-1-(\ell+i)+j]$. That is,

$$M_{F,\ell} = \begin{pmatrix} F[n-\ell-1] & F[n-\ell+0] & F[n-\ell+1] & \cdots & F[n-2] \\ F[n-\ell-2] & F[n-\ell-1] & F[n-\ell+0] & \cdots & F[n-3] \\ F[n-\ell-3] & F[n-\ell-2] & F[n-\ell-1] & \cdots & F[n-4] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ F[n-2\ell] & F[n-2\ell+1] & F[n-2\ell+2] & \cdots & F[n-\ell-1] \end{pmatrix}.$$

Observe that $M_{F,\ell}$ is a *Toeplitz* matrix (or "upside down" *Hankel* matrix) since it is constant on each descending diagonal from left to right. It follows that

$$M_{F,\ell} \times \begin{pmatrix} U_v[0] \\ U_v[1] \\ \vdots \\ U_v[\ell-1] \end{pmatrix} = \begin{pmatrix} \mathcal{A}_0 \\ \mathcal{A}_1 \\ \vdots \\ \mathcal{A}_{\ell-1} \end{pmatrix} \tag{4.1}$$

which describes a system of linear equations. Since output symbols are given modulo q, where q is assumed to be a prime, we operate in the finite field $\mathbb{Z}/q\mathbb{Z}$. It has been shown in [20] that for any ℓ, out of all the $\ell \times \ell$ Toeplitz matrices over a finite field of q elements, a fraction of exactly $(1-1/q)$ is non-singular. This fact was actually already established in [11] almost 40 years earlier but incidentally reproved in [20]. Thus, a $(1-1/q)$-fraction of all F has the property that all the ℓ input symbols in U_v can be uniquely determined from the output symbols in A_v. Since the induced distribution for U_v under any fixed $\widetilde{U}_v$ is the uniform distribution on $[q]^\ell$, the conditional entropy

$$H(A_v \mid \widetilde{U}_v = \widetilde{u}_v) \;=\; \ell \cdot \log_2 q \;\geq\; \frac{\delta \cdot \ell_v}{2},$$

where $\delta = \lceil \log_2 q \rceil$. This concludes the proof of Lemma 3.3.

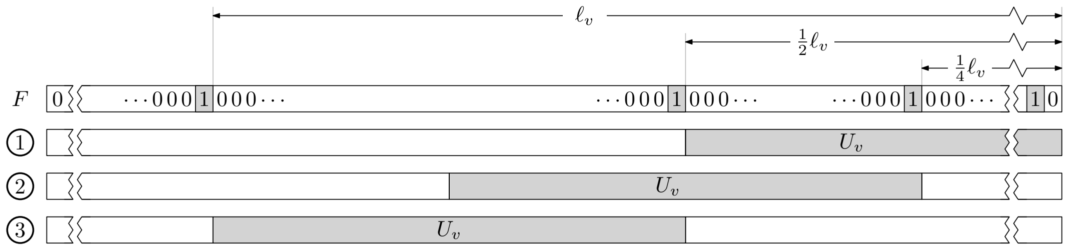

Figure 2: Three alignments of $F = K_n$ and the stream S: ① the last value of U_v has just arrived, ② half of the output symbols in A_v have been outputted, and ③ all output symbols in A_v have been outputted.

4.2 Entropy lower bound with a fixed array F

We now prove Lemma 3.4 by demonstrating that it is possible to design a fixed array F such that for all nodes $v \in \mathcal{T}$, a large portion of the values in U_v can be uniquely determined from the output symbols A_v. Since U is drawn uniformly at random from $[q]^n$, this implies large entropy of the output symbols A_v.

The fixed array F that we consider consists of stretches of 0s interspersed by 1s. The distance between two succeeding 1s is an increasing power of two, ensuring that for half of the alignments of F and S in the arrival interval where A_v is computed, all but exactly one element of U_v are simultaneously aligned with a 0 in F, hence not contributing to the outputted inner product of F and S. We define $K_n \in [2]^n$ such that

$$K_n[0], K_n[1], \ldots, K_n[n-1] \;=\; \ldots 000000000\mathbf{1}000000000000000\mathbf{1}0000000\mathbf{1}000\mathbf{1}0\mathbf{1}\mathbf{1}0\,,$$

where commas between elements on the right hand side have been omitted, or formally,

$$K_n[i] = \begin{cases} 1, & \text{if } n-1-i \text{ is a power of two;} \\ 0, & \text{otherwise.} \end{cases}$$

The hard distribution for Lemma 3.4 is $F = K_n$ and the input symbols U drawn uniformly at random from $[q]^n$.

Let v be any node of $\mathcal{T}$ and consider Figure 2 which illustrates three alignments of F and S, denoted ①, ② and ③, respectively. At alignment ①, the last value of U_v has just arrived in the stream. At alignment ②, half of the output symbols in A_v have been outputted. At alignment ③, all output symbols in A_v have been outputted. The key observation is that between alignment ② and ③, exactly one input symbol x of U_v is aligned with a 1 in F, hence x can be uniquely determined from the corresponding output symbol. Thus, over all output symbols A_v, a total of $\ell_v/4$ values of U_v can be determined, implying that the entropy of A_v must be at least $\delta \cdot \ell_v/4$, where $\delta = \lceil \log_2 q \rceil$. We now formalise this reasoning.

Using the definition of $\ell = \ell_v/2$ and the matrix $M_{F,\ell}$ above, recall that entry $M_{F,\ell}(i,j) = F[n-1-(\ell+i)+j]$. Thus, $M_{F,\ell}(i,j) = 1$ if and only if

$$n-1-\big(n-1-(\ell+i)+j\big) \;=\; \ell+i-j$$

is a power of two. Since ℓ is a power of two it follows that for row $i \in \{\ell/2, \ldots, \ell-1\}$ there can be at most one entry with the value 1. More precisely,

$$M_{F,\ell}(i,j) = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases}$$

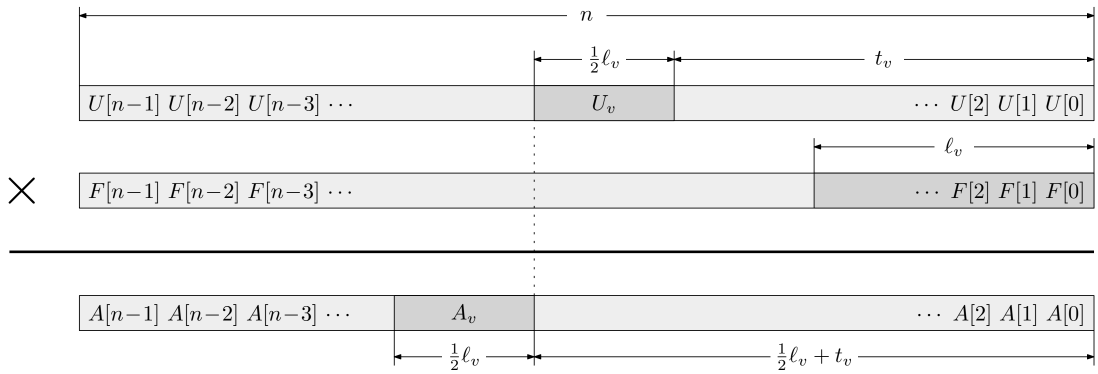

Figure 3: An illustration of $A = U \times F$. Digits of U arrive one at a time, where $U[0]$ is the low-order digit that arrives first.

From the system of linear equations in Equation (4.1) it follows that for $i \in \{\ell/2, \ldots, \ell-1\}$, $\mathcal{A}_i = U_v[i]$. Since the induced distribution for U_v under any fixed $\widetilde{U}_v$ is the uniform distribution on $[q]^{\ell}$, the conditional entropy

$$H(A_v \mid \widetilde{U}_v = \widetilde{u}_v) \;=\; \frac{\ell}{2} \cdot \log_2 q \;\geq\; \frac{\delta \cdot \ell_v}{4}\,,$$

where $\delta = \lceil \log_2 q \rceil$. This concludes the proof of Lemma 3.4.

5 Hard distributions for the multiplication problem

In this section we prove Lemmas 3.5 and 3.6, that is we show that there are instances of the online multiplication problem such that the conditional entropy of the output symbols A_v is large, where all input symbols but U_v are fixed. For the purposes of proving a lower bound we assume that all digits of the operand F are available at any time whereas the digits of the operand U arrive one at a time. Figure 3 illustrates $U \times F$, where $U[0]$ and $F[0]$ are the least significant digits and the product A is capped at n digits.

The following property of multiplying binary numbers was established by Paterson, Fischer and Meyer [31]. The lemma is stated in our notation, but the translation from the original notation of [31] is straightforward.

Lemma 5.1 (Corollary of Lemma 5 in [31]). *Suppose $q = 2$. Let v be any node of $\mathcal{T}$ and fix the digits of $\widetilde{U}_v$ arbitrarily. At least half of all $F[0, \ell_v - 1] \in [q]^{\ell_v}$ (first ℓ_v digits of F) have the property that any value of A_v can arise from at most four distinct U_v.*

Although Lemma 5.1 applies only to binary numbers, it naturally scales to any q that is a power of two. To see this, observe that the property holds for any v, and a sequence of digits in base q is after all just a bit sequence.

Corollary 5.2. *Lemma 5.1 holds for any q that is a power of two.*

We use the above corollary to prove Lemma 3.5. Let v be any node of $\mathcal{T}$. At least half of all $F \in [q^n]$ have the property that U_v can be determined up to a set of four possible values given the output symbols in A_v. Since the induced distribution for U_v under any fixed $\widetilde{U}_v$ is the uniform distribution on $[q]^\ell$ (the digits of U_v), the conditional entropy

$$H(A_v \mid \widetilde{U}_v = \widetilde{u}_v) \;\geq\; \log_2\left(\frac{q^{\ell_v/2}}{4}\right) \;\geq\; \frac{\delta \cdot \ell_v}{2} - 2\,,$$

where $\delta = \log_2 q$. This concludes the proof of Lemma 3.5.

In order to prove Lemma 3.11 we specify a fixed F which together with the uniform distribution for U gives the desired entropy lower bound. Similarly to the array K_n from Section 4.2 we define $K_{q,n}$ to be the largest number in $[q^n]$ such that the i-th bit in the binary expansion of $K_{q,n}$ is 1 if and only if i is a power of two (starting with $i = 0$ at the lower-order end). Thus, the binary expansion of $K_{q,n}$ is the reverse of $K_{n\log_2 q}$. For example, suppose that $q = 16$ (i. e., hex) and $n = 8$. Then $K_{16,8} = 10116$ in base 16, or 65,814 in decimal, since the binary expansion of $K_{16,8}$ is

$$\underbrace{0000}_{0}\underbrace{0000}_{0}\underbrace{0000}_{0}\underbrace{0001}_{1}\underbrace{0000}_{0}\underbrace{0001}_{1}\underbrace{0001}_{1}\underbrace{0110}_{6}\,.$$

Paterson, Fischer and Meyer [31] also studied the multiplication of binary numbers where one operand is fixed. The following property was given in [31], here translated into our notation.

Lemma 5.3 (Lemma 1 of [31])**.** *Suppose $q = 2$ and $F = K_{q,n}$. Let v be any node of $\mathcal{T}$ and fix the digits of $\widetilde{U}_v$ arbitrarily. Any value of A_v can arise from at most two distinct U_v.*

Similarly to Lemma 5.1 and from our definition of $K_{q,n}$, the above lemma scales to any q that is a power of two.

Corollary 5.4. *Lemma 5.3 holds for any q that is a power of two.*

We use the above corollary to prove Lemma 3.6 where $F = K_{q,n}$. Let v be any node of $\mathcal{T}$. The value of U_v can be determined up to a set of two possible values given the output symbols in A_v. Now suppose that F has this property. Since the induced distribution for U_v under any fixed $\widetilde{U}_v$ is the uniform distribution on $[q]^\ell$ (the digits of U_v), the conditional entropy

$$H(A_v \mid \widetilde{U}_v = \widetilde{u}_v) \;\geq\; \log_2\left(\frac{q^{\ell_v/2}}{2}\right) \;\geq\; \frac{\delta \cdot \ell_v}{2} - 1\,,$$

where $\delta = \log_2 q$. This concludes the proof of Lemma 3.6.

6 Hard distribution for the Hamming distance problem

In this section we prove Lemma 3.7, that is we show that there are instances of the Hamming distance problem such that the conditional entropy of the output symbols A_v is large, where all input symbols but

U_v are fixed. We will show this property for nodes in the upper part of the tree $\mathcal{T}$, namely nodes v such the number of leaves ℓ_v is greater than some constant times $\sqrt{n}$.

Unlike the hard distributions we gave for the convolution and multiplication problems, we will not give an explicit description of the array F for which the Hamming distance lower bound holds. We only show the existence of such an F. Further, for both the convolution and multiplication problems we showed that the lower bound was obtained for a majority of all F, where U was chosen uniformly at random from $[q]^n$. For the Hamming distance problem we will instead show that there exists an F and some particular subset of $[q]^n$ such that when U is drawn uniformly at random from this subset, we obtain the desired lower bound.

6.1 Terminology, choice of q and rounding issues

We will refer to the input arrays, including F and U, as *strings*, and the set $[q]$ as the *alphabet*. The values of the alphabet are referred to as *symbols*.

Unlike the convolution and multiplication problems, for the Hamming distance problem there is no benefit in having an alphabet of size greater than n, the length of F. Our hard distribution is constructed such that with an alphabet of size q, n has to be at least q^3. From now on we assume that $n \geq q^3$. Observe that whenever n is polynomial in q, the number of bits needed to represent a symbol is $\delta \in \Theta(\log n)$.

We will often treat various roots of integers as integers. For example, we may say that some string of length $q^{3/2}$ is the concatenation of q smaller strings, each of length $q^{1/2}$. This is of course only possible whenever these numbers are integers, which is not the case for arbitrary q. One could overcome this problem by adjusting the values with appropriate floors and ceilings, as well as introducing padding symbols where necessary, but this would without doubt clutter the presentation. We have decided to keep it simple by treating any root of any integer as an integer, and assuming that everything adds up nicely. This is only to keep the presentation clean and it should be obvious from the context that this has no impact on the asymptotic behaviour.

6.2 The overall structure of the fixed string F

Recall the definition of the array $K_n \in \{0,1\}^n$ from Section 4.2 which consists of 0s everywhere except at power-of-two positions from the right-hand end. A hard distribution for the convolution problem was given by setting F to K_n and choosing U uniformly at random from $[q]^n$. Recall Figure 2 which illustrates why we chose this hard distribution: for each output symbol in the second half of A_v, that is between the alignments marked ② and ③ in the figure, exactly one input symbol of U_v is aligned with a 1 in F and all other input symbols of U_v are aligned with 0. Thus, the second half of U_v can be uniquely determined from the output symbols A_v.

To show a lower bound for the Hamming distance problem we devise a string F that resembles K_n. First we introduce an auxiliary string R of length $\Theta(q^{3/2})$. We will use $r = (q/2)^{3/2}$ as a shorthand for the length of R. We will give the details of R later but will highlight an important property of it below. We obtain F from K_n by first replacing each 0 by a symbol that we denote $\star$. The symbol $\star$ will never occur in the stream, hence will always generate a mismatch. We then replace every length-r substring starting at a 1 with a copy of R. Any 1 that is closer than r positions from the right-hand end of F is replaced by a $\star$-symbol instead. Figure 4 illustrates F.

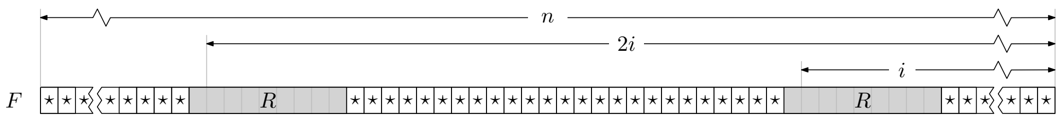

Figure 4: The string F has a copy of R starting at each position $n-1-i$ where $i \geq |R|$ is a power of two. All other positions have the symbol $\star$ which only occurs in F and not in the stream.

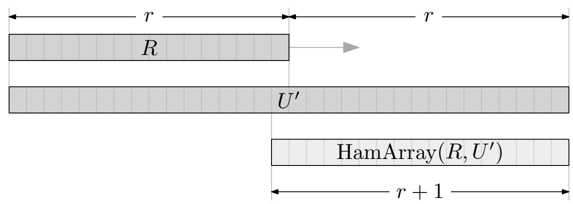

Figure 5: $\mathrm{HamArray}(R,U')$ contains the Hamming distances between R and every length-r substring of U' as R slides along U'.

6.3 Properties of the string R and Hamming arrays

The string R will play the same role as the value 1 in K_n did for the convolution problem, namely it will allow us to uniquely determine symbols from U. To see how, we first introduce the notion of a Hamming array, illustrated in Figure 5. For a string U' of length $2r$, we write $\mathrm{HamArray}(R,U')$ to denote the length-$(r+1)$ array such that for $i \in [r+1]$, $\mathrm{HamArray}(R,U')[i]$ is the Hamming distance between R and $U'[i,i+r-1]$. That is, $\mathrm{HamArray}(R,U')$ contains the Hamming distances between R and every length-r substring of U'.

To see the resemblance with a 1 in K_n, we give the following lemma. The proof is non-trivial and deferred to Section 7.3. A high-level explanation of the lemma is given immediately after its statement.

Lemma 6.1. *There exists a constant $k > 0$ such that for any r there is a length-r string $R \in [q]^r$ where $q = 2r^{2/3}$ such that*

$$\left| \left\{ \mathrm{HamArray}(R,U') \;\middle|\; U' \in [q]^{2r} \right\} \right| \geq q^{kr}.$$

The lemma says that there is a string R such that over all possible U' of length $2|R|$, one can obtain $q^{\Theta(r)}$ distinct Hamming arrays. Since there are only q^{2r} possible values of U', this means that a non-negligible fraction of all U' can be put in one-to-one correspondence with Hamming arrays. Thus, as symbols in U_v slide past an R in a similar fashion to symbols in U_v sliding past a 1 in K_n in the hard distribution for the convolution problem, we can infer a substantial portion of the symbols of U_v from the output symbols A_v, hence obtain large entropy. We formalise this in the next section and explain how the lower bound is obtained.

6.4 The hard distribution and obtaining the lower bound

Relying on Lemma 6.1 above we will now describe a hard distribution for the Hamming distance problem and use it to prove Lemma 3.7, which is the purpose of Section 6.

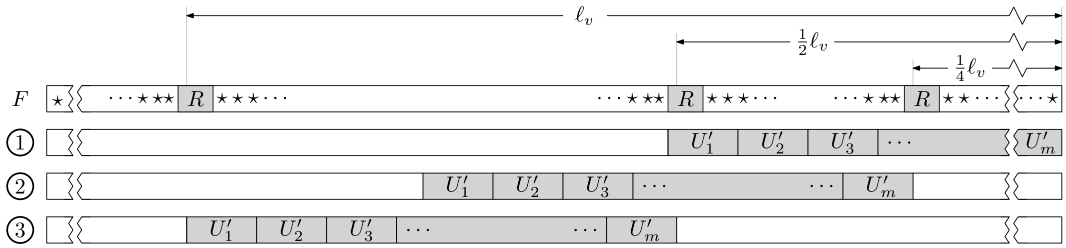

Figure 6: Three alignments of F and the stream S: ① the last value of U_v has just arrived, ② half of the outputs in A_v have been outputted, and ③ all outputs in A_v have been outputted. The string U_v is here the concatenation of $U'_1, \ldots, U'_m \in \mathcal{U}_R$, where $m = \ell_v/(4r)$.

Given a string $R \in [q]^r$, we let $\mathcal{U}_R \subseteq [q]^{2r}$ be any largest set of length-$(2r)$ strings such that for any two distinct strings $U'_1, U'_2 \in \mathcal{U}_R$,

$$\text{HamArray}(R, U'_1) \neq \text{HamArray}(R, U'_2)\,.$$

To uniquely specify a string in $\mathcal{U}_R$ we need $\log_2 |\mathcal{U}_R|$ bits. By Lemma 6.1 we have that there exists an R such that $\log_2 |\mathcal{U}_R| \in \Theta(r \log q)$.

For the hard distribution we use F from above with an R that has the properties of Lemma 6.1. The input U is given by concatenating $n/2r$ strings drawn independently and uniformly at random from $\mathcal{U}_R$.

Similarly to Figure 2 we can now illustrate how strings from $\mathcal{U}_R$ slide past R during the second half of the output symbols in A_v, where v is any node of $\mathcal{T}$ such that $\ell_v \geq h \cdot \sqrt{n}$. Here h is the positive constant from Lemma 3.7. We defer picking the value of h until later. For any such node, we have that $\ell_v > h \cdot r$ because we assumed that $n \geq q^3$ and the definition of r implies that $q^3 > r^2$. In Figure 6 we have illustrated U_v as the concatenation of random strings $U'_1, \ldots, U'_m$ drawn from $\mathcal{U}_R$, where $m = \ell_v/(4r)$. Between alignments ② and ③ in the figure, the second half of the substrings U'_i of U_v slide in turn past R, and from the outputs in A_v we can infer $\text{HamArray}(R, U'_i)$ for each such U'_i. By construction of $\mathcal{U}_R$ this allows us to uniquely determine the strings U'_i. Thus, over all outputs A_v, a total of approximately $m/2$ substrings U'_i of U_v can be completely determined. We pick above h to be sufficiently large so that, even compensating for border cases, the number of substrings U'_i of U_v that can be determined is always at least one. This implies that the entropy of A_v must, by Lemma 6.1, be at least $\Theta((m/2) \cdot r \log q) = \Theta(\ell_v \cdot \delta)$, where $\delta = \lceil \log_2 q \rceil$. This concludes the proof of Lemma 3.7.

7 A string with many different Hamming arrays

In this section we prove Lemma 6.1, that is we show that there exists a string R which gives many different Hamming arrays. This is arguably the most technically detailed part of our lower bound proofs. To recap, we claim that for any r there exists a string $R \in [q]^r$ with $q = 2r^{2/3}$ which permits at least q^{kr} distinct Hamming arrays when combined with every string in $[q]^{2r}$, where k is a constant. Next we describe the overall structure of an R with this property.

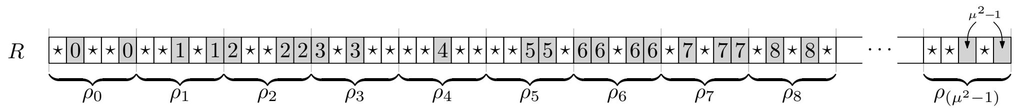

Figure 7: An example of the string R of length $r = \mu^3$, which is the concatenation of the μ^2 strings $\rho_0, \ldots, \rho_{\mu^2-1}$, where $\rho_i \in \{\star, i\}^\mu$.

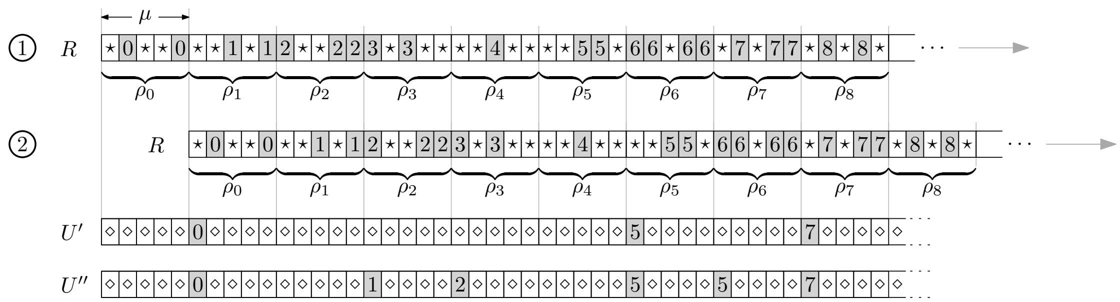

Figure 8: Setting symbols of U' renders a large set of possible Hamming distance outputs.

7.1 The structure of R

To shorten notation it will be convenient to introduce the variable μ as a shorthand for $r^{1/3}$. Hence R has length $r = \mu^3$ and $q = 2\mu^2$. The string R is constructed by concatenating μ^2 substrings, each of length μ. For $i \in [\mu^2]$ we let ρ_i denote the i-th substring of R, that is

$$R = \rho_0 \rho_1 \cdots \rho_{(\mu^2-1)} .$$

Each substring ρ_i can only contain symbols from the set $\{\star, i\}$, where $\star$ is the special symbol that will not occur in the stream. Therefore, the total number of distinct symbols in R is at most $\mu^2 + 1 \leq q$ as claimed. Figure 7 illustrates an example of R.

The purpose of the substrings ρ_i is to support a reduction from vector addition to Hamming arrays that we explain next.

7.2 Vector sums and Hamming arrays

Before we describe the full reduction in the following section we begin by giving some high level intuition. We first introduce some notation. Let $V = \{v_0, v_1, v_2 \ldots v_{\mu^2-1}\}$ be a multi-set of vectors where $v_i \in \{0,1\}^\mu$. We can define a correspondence between the length-μ substring ρ_i of R and the i-th vector v_i as follows. The j-th symbol of ρ_i equals i if the j-th component of v_i is 1 and $\star$ otherwise. For example, $\rho_2 = 2\star\star22$ from Figure 7 corresponds to the vector $v_2 = (1,0,0,1,1)$.

We will now see that this correspondence between vectors in V and substrings of R can be used to encode elementwise *sums* of vectors in V in the Hamming array for R. Consider the string $U' \in [q]^{2r}$ in Figure 8 as an illustrative example. Here the string U' contains the other special symbol that we introduce, denoted $\diamond$. This symbol does not occur in R, hence will always mismatch. In the figure we see that all positions of U' have the symbol $\diamond$, except for three positions where the symbols are 0, 5 and 7, respectively. The positions holding these symbols are chosen such that in the first alignment between R and U', marked ①, the symbols 0, 5 and 7 sit immediately after ρ_0, ρ_5 and ρ_7 in R, respectively. As R slides μ steps to the right towards the alignment marked ②, the symbols 0, 5 and 7 of U' will generate matches whenever they are aligned with their corresponding symbols in R. Thus, for $i \in \{1,\dots,\mu\}$,

$$\mathrm{HamArray}(R,U')[i] \;=\; r-(v_0+v_5+v_7)[i]\,,$$

where $(v_0+v_5+v_7)[i]$ is the i-th component of the sum of the vectors v_0, v_5 and v_7. In other words, from $\mathrm{HamArray}(R,U')[1,\mu]$ we can uniquely determine the sum $v_0+v_5+v_7$. As shorthand we will think of $\mathrm{HamArray}(R,U')[1,\mu]$ as *encoding* the sum $v_0+v_5+v_7$.

The idea above can be repeated by populating U' with more symbols from $[\mu^2]$. As an example we have added the symbols 1 and 2, and another copy of 5 to U', which is the string denoted U'' in the figure. As R slides another μ steps to the right, $\mathrm{HamArray}(R,U'')[\mu+1,2\mu]$ encodes (uniquely specifies) the sum $v_1+v_2+v_5$.

Observe that as we populate U' with symbols, positions become *blocked*. For example, we could not have added symbols to U' to define an alternate U'' which instead encodes the sum $v_1+v_2+v_4$ in $\mathrm{HamArray}(R,U'')[\mu+1,2\mu]$ since the position where the 4 would have to be set is already occupied by a 5. Observe however that setting symbols of U'' as above generates matches only in the intended length-μ window of the Hamming array. Thus, we have full control of which vector sums we want to encode, under the constraint that positions become blocked, limiting the choice of vectors.

The conclusion thus far is that vector sums have a direct correspondence with the Hamming array. Next we take the ideas from above further and show that if there exists a pool of μ^2 vectors such that many different vector sums can be obtained when adding μ vectors from the pool, then the number of distinct $\mathrm{HamArray}(R,U')$ one can obtain is large. This would prove Lemma 6.1.

7.3 The string R and the proof of Lemma 6.1

Before we state the next lemma which will be used to prove Lemma 6.1, we introduce some basic notation that we will use when reasoning about multisets of vectors. Let X be a multiset of vectors. Consider an arbitrary ordering of the elements of X and refer to $X[i]$ as the i-th element of X. We use the term *sub-multiset* of X to denote any multiset obtained from X by removing zero or more elements. We will use the notation $\sqsubseteq$ to denote the sub-multiset relation so that we have, for example, $\{1,1,4,5,5\} \sqsubseteq \{1,1,1,4,4,5,5,7,8\}$. We can now introduce the required lemma.

Lemma 7.1. *For any $\mu > 40$ such that $\mu-1$ is a prime, there exists a multiset V of vectors from $\{0,1\}^\mu$ such that $|V| = \mu(\mu-1)$ and for any sub-multiset $V' \subseteq V$ of size at least $(63/64)|V|$,*

$$\Big|\{\, w_1+\cdots+w_\mu \;\mid\; \{w_1,\dots,w_\mu\} \sqsubseteq V' \,\}\Big| \;\geq\; \mu^{(\mu/10)}\,.$$

The lemma is proved in Section 8 and we will now use it to construct an R that proves Lemma 6.1. The introduction of a sub-multiset V' in the lemma above is to reflect the fact that positions of U' get blocked as we populate it with symbols. We will see next that at any step, a fraction of at most $1/64$ of the μ^2 vectors are blocked.

Suppose that $V = \{v_0, \dots, v_{\mu(\mu-1)}\}$ is a multiset of length-μ vectors over $\{0,1\}$ with the properties of Lemma 7.1. That is, we assume that $\mu > 40$ and $\mu - 1$ is a prime. Again as discussed in Section 6.1, we can always tweak relevant values in order to meet this criteria.

The string R is simply chosen such that for $i \in [\mu(\mu-1)]$, the substring ρ_i corresponds to the vector v_i of V as described at the start of Section 7.2. For $i \in \{\mu(\mu-1), \dots, (\mu^2-1)\}$, the substring $\rho_i = \{\star\}^\mu$ as we will ignore these substrings anyway. In order to show that this R proves Lemma 6.1 we will populate a length-$(2r)$ vector U' with symbols and show how length-μ subarrays of $\mathrm{HamArray}(R,U')$ correspond to vector sums of μ vectors chosen arbitrarily from a sub-multiset of V. The process for constructing a string U' is as follows:

1. Set all $2\mu^3$ positions of U' to the symbol $\diamond$.

2. Align R with the left half of U' as illustrated in Figure 5.

3. Let $V' \subseteq V$ be the set of vectors that are not blocked. Initially this means that $V' = V$ but as we return to this step, V' shrinks.

4. Choose any sub-multiset $\{w_1, \dots, w_\mu\} \sqsubseteq V'$ and set their corresponding positions in U' accordingly. For any $v_i \in \{w_1, \dots, w_\mu\}$ the corresponding position in U' is the one which is currently aligned with the position immediately after substring p_i in R. This position is set to the symbol i.

5. Slide R by μ steps along U'. Over these alignments, $\mathrm{HamArray}(R,U')$ uniquely specifies the vector sum $w_1 + \cdots + w_\mu$.

   Steps 3–5 are referred to as a *round*.

6. Repeat from Step 3 for a total of $(\mu-1)/64$ rounds. Observe that a total of $\mu(\mu-1)/64 = (1/64)|V|$ vectors become blocked, hence $|V'|$ is always at least $(63/64)|V|$.

7. Slide R by *one single step* along U'. This will offset all previously blocked vectors and allow us to start over again at Step 3 as if no vectors are blocked. This is repeated until this step is reached for the μ-th time. At that point the offsetting of blocked vectors has cycled and previously set positions of U' are yet again blocked.

Populating U' according to the procedure above means that R is shifted by a total of

$$\mu \cdot (\mu-1)/64 \cdot \mu + (\mu - 1) \;=\; \mu^3/64 - \mu^2/64 + \mu - 1 \;<\; r$$

steps. Over these steps we have by Lemma 7.1 that for each length-μ subarray of $\mathrm{HamArray}(R,U')$ that corresponds to a vector sum, there is a choice of at least $\mu^{(\mu/10)}$ distinct values. Thus, when $\mu > 40$, the number of distinct $\mathrm{HamArray}(R,U')$ is at least

$$\left(\mu^{(\mu/10)}\right)^{\mu(\mu-1)/64} \;=\; \mu^{(\mu^3-\mu^2)/640} \;\geq\; \mu^{(\mu^3/656)} \;=\; \left(r^{(1/3)}\right)^{(r/656)} \;=\; r^{kr},$$

where $k = 1/1968$. This concludes the proof of Lemma 6.1.

8 Vector sets with many distinct sums

In this section, we prove Lemma 7.1. We first rephrase the lemma slightly by introducing some notation. For any multiset V' of vectors from $\{0,1\}^\mu$, we define

$$\mathrm{Sum}(V') = \Big|\{\, w_1 + \cdots + w_\mu \mid \{w_1, \ldots, w_\mu\} \sqsubseteq V' \,\}\Big|$$

to be the size of the set of distinct vector sums one can obtain by summing the vectors of size-μ sub-multisets of V'. Addition is elementwise and over the integers. Lemma 7.1 says that there exists a multiset V of vectors from $\{0,1\}^\mu$ such that $|V| = \mu(\mu-1)$ and for any sub-multiset $V' \sqsubseteq V$ of size at least $(63/64)|V|$, we have that $\mathrm{Sum}(V') \geq \mu^{(\mu/10)}$.

Our approach will be an application of the probabilistic method. Specifically, we will show that when the vectors of V are chosen independently and uniformly at random from $\{0,1\}^\mu$, the expected value,

$$\mathbb{E}\,[\mathrm{Sum}(V)] \geq \frac{1}{2}(\mu-1)^{(\mu/9)}.$$

Thus, there must exist a V such that $\mathrm{Sum}(V) \geq (\mu-1)^{(\mu/9)}/2$. Given such a V, we then show that for every sub-multiset $V' \sqsubseteq V$ such that $|V'| \geq (63/64)|V|$, $\mathrm{Sum}(V') \geq \mu^{(\mu/10)}$.

8.1 Vectors and codes

We now describe a connection between vectors and codes which we will use in our analysis to lower bound the number of distinct vector sums, $\mathrm{Sum}(V)$ that can be obtained from a vector set V. We will require the following lemma from the field of Coding Theory. The lemma is tailored for our needs and is a special case of "Construction II" in [1]. For our purposes, a binary constant-weight constant-weight binary cyclic code can be seen simply as set of bit strings (codewords) with two additional properties: the first is that all codewords have constant Hamming weight μ, i. e., they have exactly μ 1s, and the second property is that any cyclic shift of a codeword is also a codeword.

Lemma 8.1 ([1]). *For any $\mu \geq 4$ such that $\mu - 1$ is a prime and any odd $\gamma \in [\mu]$, there is a binary constant-weight constant-weight binary cyclic code with $(\mu-1)^\gamma$ codewords of length $\mu(\mu-1)$ and Hamming weight μ such that any two codewords have Hamming distance at least $2(\mu-\gamma)$.*

Let $\widetilde{C}$ be the binary code that contains *all* codewords of length $\mu(\mu-1)$ with Hamming weight μ. We can think of a codeword of $\widetilde{C}$ representing a size-μ sub-multiset $X \sqsubseteq V$ such that the i-th vector of V (under any enumeration of the elements of V) is in X if and only if position i of the codeword is 1. That is, $\widetilde{C}$ represents all possible sub-multisets of V of size μ. To shorten notation, we refer to $\widetilde{c} \in \widetilde{C}$ as both a codeword and a sub-multiset of μ vectors from V.

Suppose that $\mu \geq 4$ and $\mu - 1$ is a prime. We let $C \subseteq \widetilde{C}$ be a cyclic code of size $(\mu-1)^\gamma$, where γ is any odd integer in the interval $[\mu/9, \mu/8]$, such that the Hamming distance between any two codewords in C is at least $7\mu/4$. The existence of such a C is guaranteed by Lemma 8.1 since $2(\mu - \mu/8) = 7\mu/4$. Observe that every codeword of C has Hamming weight μ.

For $c \in C$ we define the *ball*, $\mathrm{Ball}(c)$ to be the set of bit strings in $\widetilde{C}$ at Hamming distance at most $\mu/16$ from c. Formally,

$$\mathrm{Ball}(c) = \{\, \widetilde{c} \mid \widetilde{c} \in \widetilde{C} \text{ and Hamming distance between } c \text{ and } \widetilde{c} \text{ is at most } \mu/16 \,\} .$$

Observe that the $|C|$ balls are all disjoint since the Hamming distance between any two codewords in C is at least than $7\mu/4$. In particular, for any $c \in C$, $\mathrm{Ball}(c) \cap C = \{c\}$. We have that for any $c \in C$, using the fact $\binom{a}{b} \leq (ae/b)^b$,

$$\left|\mathrm{Ball}(c)\right| \leq \binom{\mu}{\mu/16} \cdot \binom{|V|}{\mu/16} \leq \left(\frac{\mu e \cdot |V| e}{(\mu/16)^2}\right)^{\mu/16} \leq \left(\frac{\mu}{16}\right)^{\mu/16} .$$

For $\widetilde{c} \in \widetilde{C}$ we write $\mathrm{sum}(\widetilde{c})$ to denote the vector in $[\mu+1]^\mu$ obtained by adding the μ vectors in the vector set $\widetilde{c}$, that is $\mathrm{sum}(\widetilde{c})$ vector sum of the vectors represented by $\widetilde{c}$.

Towards proving Lemma 7.1 we will show that when the vectors of V are chosen uniformly at random, we expect more than half of all $|C|$ balls to have the property that for every $\widetilde{c}$ in the ball, $\mathrm{sum}(\widetilde{c})$ can only be obtained by summing vectors from that ball.

8.2 Choosing the vectors in V

So far we have not discussed the choice of vectors in V. Initially we consider choosing the vectors independently and uniformly at random from $\{0,1\}^\mu$. We will first show that

$$\mathbb{E}\left[\mathrm{Sum}(V)\right] \geq \frac{1}{2}(\mu-1)^{(\mu/9)} ,$$

then we will fix V and show that this fixed V has the property of Lemma 7.1.

Consider any two distinct $c_1, c_2 \in C$ and the corresponding balls, $\mathrm{Ball}(c_1)$ and $\mathrm{Ball}(c_2)$ which are disjoint subsets of $\widetilde{C}$. For any $\widetilde{c_1} \in \mathrm{Ball}(c_1)$ and $\widetilde{c_2} \in \mathrm{Ball}(c_2)$, we now analyse the probability that $\mathrm{sum}(\widetilde{c_1}) = \mathrm{sum}(\widetilde{c_2})$. From the definitions above it follows that $\widetilde{c_1}$ and $\widetilde{c_2}$ must differ on at least $7\mu/4 - 2(\mu/16) \geq \mu$ positions, implying that the two vector sets $\widetilde{c_1}$ and $\widetilde{c_2}$ have at most $\mu/2$ vectors in common, thus at least $\mu/2$ of the vectors in $\widetilde{c_1}$ are not in $\widetilde{c_2}$. Let $w_1, \ldots, w_{(\mu/2)}$ denote an arbitrary choice of $\mu/2$ of those vectors. For $i \in [\mu]$ we can write the i-th component of $\mathrm{sum}(\widetilde{c_1})$ as

$$\mathrm{sum}(\widetilde{c_1})[i] \;=\; w_1[i] + \cdots + w_{(\mu/2)}[i] + x[i] ,$$

where the vector x does not depend on $w_1, \ldots, w_{(\mu/2)}$. In order to have $\mathrm{sum}(\widetilde{c_1}) = \mathrm{sum}(\widetilde{c_2})$ we must have

$$w_1[i] + \cdots + w_{(\mu/2)}[i] \;=\; \mathrm{sum}(\widetilde{c_2})[i] - x[i]$$

for each $i \in [\mu]$. Since the vectors are picked independently and uniformly at random from $\{0,1\}^\mu$, the most likely value of $w_1[i] + \cdots + w_{(\mu/2)}[i]$ is $\mu/4$. The probability that this sum equals $\mu/4$ is

$$\Pr\left(w_1[i] + \cdots + w_{(\mu/2)}[i] = \frac{\pi}{4}\right) \;=\; \binom{\mu/2}{\mu/4} \cdot \left(\frac{1}{2}\right)^{\mu/2} \;\leq\; \left(\frac{\mu}{2}\right)^{-1/2} ,$$

where the inequality follows from the fact that for any a, $\binom{a}{a/2} \leq 2^a/\sqrt{a}$. Thus, the probability that $\mathrm{sum}(\widetilde{c_1}) = \mathrm{sum}(\widetilde{c_2})$, that is $\mathrm{sum}(\widetilde{c_1})[i] = \mathrm{sum}(\widetilde{c_2})[i]$ for all $i \in [\mu]$, is

$$\Pr\big(\mathrm{sum}(\widetilde{c_1}) = \mathrm{sum}(\widetilde{c_2})\big) \leq \left(\left(\frac{\mu}{2}\right)^{-1/2}\right)^{\mu} \leq \left(\frac{\mu}{2}\right)^{-\mu/2}. \tag{8.1}$$

For two distinct $c_1, c_2 \in C$, we define the indicator random variable

$$I(c_1,c_2) = \begin{cases} 0 & \text{if } \mathrm{sum}(\widetilde{c_1}) = \mathrm{sum}(\widetilde{c_2}) \text{ for some } \widetilde{c_1} \in \mathrm{Ball}(c_1) \text{ and } \widetilde{c_2} \in \mathrm{Ball}(c_2), \\ 1 & \text{otherwise.} \end{cases}$$

Taking the union bound over all $\widetilde{c_1} \in \mathrm{Ball}(c_1)$ and $\widetilde{c_2} \in \mathrm{Ball}(c_2)$, and using the probability bound in Equation (8.1), we have

$$\begin{aligned} \Pr\big(I(c_1,c_2) = 0\big) &\leq \big|\mathrm{Ball}(c_1)\big| \cdot \big|\mathrm{Ball}(c_2)\big| \cdot \left(\frac{\mu}{2}\right)^{-\mu/2} \\ &\leq \left(\frac{\mu}{16}\right)^{2(\mu/16)} \left(\frac{\mu}{2}\right)^{-\mu/2} \\ &\leq \left(\frac{1}{\mu^3}\right)^{\mu/8}. \end{aligned} \tag{8.2}$$

For any $c_1 \in C$, we now define the indicator random variable

$$I'(c_1) = \begin{cases} 0 & \text{if } I(c_1,c_2) = 0 \text{ for some } c_2 \in C \setminus \{c_1\}, \\ 1 & \text{otherwise.} \end{cases}$$

That is, $I'(c_1) = 1$ if and only if $\mathrm{sum}(\widetilde{c_1}) \neq \mathrm{sum}(\widetilde{c_2})$ for every $\widetilde{c_1} \in \mathrm{Ball}(c_1)$ and every $\widetilde{c_2}$ in a different ball. In other words, each sums of codewords, $\mathrm{sum}(\widetilde{c_1})$ corresponding to a $\widetilde{c_1} \in \mathrm{Ball}(c_1)$ is unique to this ball. It is possible, however that $\mathrm{sum}(\widetilde{c_1}) = \mathrm{sum}(\widetilde{c_2})$ if $\widetilde{c_2}$ is in the same ball as $\widetilde{c_1}$. We say that $\mathrm{Ball}(c_1)$ is *good* if and only if $I'(c_1) = 1$.

Taking the union bound over all $c_2 \in C$, and using Equation (8.2) and the fact that $|C| \leq \mu^{(\mu/8)}$, we have

$$\begin{aligned} \Pr\big(I'(c_1) = 0\big) &\leq \sum_{c_2 \in C \setminus \{c_1\}} \Pr\big(I(c_1,c_2) = 0\big) \\ &\leq |C| \left(\frac{1}{\mu^3}\right)^{\mu/8} \leq \mu^{(\mu/8)} \left(\frac{1}{\mu^3}\right)^{\mu/8} \leq \frac{1}{2}. \end{aligned}$$

By linearity of expectation we have that the expected number of good balls is

$$\mathbb{E}\left[\sum_{c \in C} I'(c)\right] \geq \frac{|C|}{2}.$$

The conclusion is that there is a multiset V of vectors for which at least $|C|/2$ balls are good. As $|C| \geq (\mu-1)^{(\mu/9)}$ it then follows that as required,

$$\mathrm{Sum}(V) \geq \frac{1}{2}(\mu-1)^{(\mu/9)}.$$

8.3 Many distinct sums for subsets of V

Let V be a multiset of vectors such that the number of good balls is at least $|C|/2$ in accordance with the conclusion of the previous section. As observed above, $\mathrm{Sum}(V) \geq (\mu-1)^{(\mu/9)}/2$. It remains to show that for any sub-multiset $V' \sqsubseteq V$ of size $(63/64)|V|$, $\mathrm{sum}(V')$ is also large.

Over all codewords in C, seen as bit strings, the total number of 1s is $|C|\mu$. Since C is cyclic, the number of codewords in C that have a 1 in position $i \in [|V|]$ is the same as the number of codewords that have a 1 in position j, for any $j \in [|V|]$. Thus, for each one of the $|V|$ positions there are exactly $|C|\mu/|V|$ codewords in C with a 1 in that position.

Let $V' \sqsubseteq V$ be of size $(63/64)|V|$. Let J be the set of $|V|/64$ positions that correspond to the vectors of V that are not in V'. We will now modify the codewords of C as follows. For each $j \in J$ and codeword $c \in C$ we set $c[j]$ to 0. The total number of 1s across all codewords in C is therefore reduced from $|C|\mu$ by exactly

$$\frac{|V|}{64} \cdot \frac{|C|\mu}{|V|} = \frac{|C|\mu}{64}\,.$$

The number of codewords of C that have lost $\mu/16$ or more 1s is therefore at most

$$\frac{|C|\mu}{64} \Big/ \frac{\mu}{16} = \frac{|C|}{4}\,.$$

Let $C' \subseteq C$ be the set of codewords c that have lost fewer than $\mu/16$ 1s and for which $\mathrm{Ball}(c)$ is good. Since there are at least $|C|/2$ good balls, $|C'| \geq |C|/4$.

Let the code C'' be obtained from C' by replacing, for each codeword in C', every removed 1 with a 1 at some other arbitrary position that is not in J. Thus, every $c'' \in C''$ has Hamming weight μ and belongs to the good ball $\mathrm{Ball}(c')$, where c'' was obtained from $c' \in C'$. Hence $|C''| = |C'| \geq |C|/4$. Every codeword of $c'' \in C''$ corresponds to a sub-multiset of V of size μ. Crucially, this sub-multiset only contains vectors from the sub-multiset V'. Further, from the definition of a good ball we have that each codeword in C'' corresponds to a sub-multiset of V' with a distinct vector sum. As $|C''| \geq |C|/4$ we have that at least $|C|/4$ distinct vector sums can be obtained by adding μ vectors from V'. Thus,

$$\mathrm{Sum}(V') \geq \frac{1}{4}(\mu-1)^{(\mu/9)} \geq \mu^{(\mu/10)}$$

when $\mu > 40$. This completes the proof of Lemma 7.1.

References

[1] NGUYEN QUANG A, LÁSZLÓ GYŐRFI, AND JAMES L. MASSEY: Constructions of binary constant-weight cyclic codes and cyclically permutable codes. *IEEE Trans. Inform. Theory*, 38(3):940–949, 1992. [doi:10.1109/18.135636] 24

[2] KARL R. ABRAHAMSON: Generalized string matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987. [doi:10.1137/0216067] 4

[3] AMIHOOD AMIR, MOSHE LEWENSTEIN, AND ELY PORAT: Faster algorithms for string matching with k mismatches. *J. Algorithms*, 50(2):257–275, 2004. Preliminary version in SODA'00. [doi:10.1016/S0196-6774(03)00097-X] 4

[4] JOSHUA BRODY, AMIT CHAKRABARTI, ODED REGEV, THOMAS VIDICK, AND RONALD DE WOLF: Better gap-Hamming lower bounds via better round elimination. In *Proc. 14th Internat. Workshop on Randomization and Computation (RANDOM'10)*, pp. 476–489, 2010. [doi:10.1007/978-3-642-15369-3_36, arXiv:0912.5276] 2

[5] AMIT CHAKRABARTI AND ODED REGEV: An optimal lower bound on the communication complexity of gap-Hamming-distance. *SIAM J. Comput.*, 41(5):1299–1317, 2012. Preliminary version in STOC'11. [doi:10.1137/120861072] 2

[6] RAPHAËL CLIFFORD, KLIM EFREMENKO, BENNY PORAT, AND ELY PORAT: A black box for online approximate pattern matching. *Inform. and Comput.*, 209(4):731–736, 2011. Preliminary version in CPM'08. [doi:10.1016/j.ic.2010.12.007] 3, 4, 5

[7] RAPHAËL CLIFFORD AND MARKUS JALSENIUS: Lower bounds for online integer multiplication and convolution in the cell-probe model. In *Proc. 38th Internat. Colloq. on Automata, Languages and Programming (ICALP'11)*, pp. 593–604. Springer, 2011. [doi:10.1007/978-3-642-22006-7_50, arXiv:1101.0768] 1

[8] RAPHAËL CLIFFORD, MARKUS JALSENIUS, AND BENJAMIN SACH: Tight cell-probe bounds for online Hamming distance computation. In *Proc. 24th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'13)*, pp. 664–674. ACM Press, 2013. [doi:10.1137/1.9781611973105.48, arXiv:1207.1885] 1

[9] RAPHAËL CLIFFORD AND BENJAMIN SACH: Pattern matching in pseudo real-time. *J. Discrete Algorithms*, 9(1):67–81, 2011. Preliminary version in CPM'10. [doi:10.1016/j.jda.2010.09.005] 3

[10] GRAHAM CORMODE, MAYUR DATAR, PIOTR INDYK, AND S. MUTHUKRISHNAN: Comparing data streams using Hamming norms (how to zero in). *IEEE Trans. on Knowl. and Data Eng.*, 15(3):529–540, 2003. Preliminary version in VLDB'02. [doi:10.1109/TKDE.2003.1198388] 2

[11] DAVID E. DAYKIN: Distribution of bordered persymmetric matrices in a finite field. *J. reine angew. Math.*, 203:47–54, 1960. [doi:10.1515/crll.1960.203.47] 14

[12] MICHAEL J. FISCHER AND LARRY J. STOCKMEYER: Fast on-line integer multiplication. *J. Comput. System Sci.*, pp. 317–331, 1974. Preliminary version in STOC'73. [doi:10.1016/S0022-0000(74)80047-4] 3, 4

[13] MICHAEL L. FREDMAN: Observations on the complexity of generating quasi-Gray codes. *SIAM J. Comput.*, 7(2):134–146, 1978. [doi:10.1137/0207012] 5

[14] MICHAEL L. FREDMAN AND MIKE E. SAKS: The cell probe complexity of dynamic data structures. In *Proc. 21st STOC*, pp. 345–354, 1989. [doi:10.1145/73007.73040] 5

[15] ZVI GALIL: String matching in real time. *J. ACM*, 28(1):134–149, 1981. [doi:10.1145/322234.322244] 3

[16] TORBEN HAGERUP: Sorting and searching on the word RAM. In *Proc. 15th Symp. Theoretical Aspects of Comp. Sci. (STACS'98)*, pp. 366–398, 1998. [doi:10.1007/BFb0028575] 5

[17] WEI HUANG, YAOYUN SHI, SHENGYU ZHANG, AND YUFAN ZHU: The communication complexity of the Hamming distance problem. *Inform. Process. Lett.*, 99(4):149–153, 2006. [doi:10.1016/j.ipl.2006.01.014] 2

[18] PIOTR INDYK: Faster algorithms for string matching problems: Matching the convolution bound. In *Proc. 39th FOCS*, pp. 166–173, 1998. [doi:10.1109/SFCS.1998.743440] 4

[19] T. S. JAYRAM, RAVI KUMAR, AND D. SIVAKUMAR: The one-way communication complexity of Hamming distance. *Theory of Computing*, 4(6):129–135, 2008. [doi:10.4086/toc.2008.v004a006] 2

[20] ERICH KALTOFEN AND AUSTIN A. LOBO: On rank properties of Toeplitz matrices over finite fields. In *Proc. International Symp. on Symbolic and Algebraic Computation (ISSAC'96)*, pp. 241–249, 1996. [doi:10.1145/236869.237081] 14

[21] HOWARD J. KARLOFF: Fast algorithms for approximately counting mismatches. *Inform. Process. Lett.*, 48(2):53–60, 1993. [doi:10.1016/0020-0190(93)90177-B] 4

[22] S. RAO KOSARAJU: Efficient string matching. Manuscript, 1987. 4

[23] GAD M. LANDAU AND UZI VISHKIN: Efficient string matching with k mismatches. *Theoret. Comput. Sci.*, 43:239–249, 1986. [doi:10.1016/0304-3975(86)90178-7] 4

[24] KASPER GREEN LARSEN: The cell probe complexity of dynamic range counting. In *Proc. 44th STOC*, pp. 85–94, 2012. [doi:10.1145/2213977.2213987, arXiv:1105.5933] 5

[25] KASPER GREEN LARSEN, OMRI WEINSTEIN, AND HUACHENG YU: Crossing the logarithmic barrier for dynamic Boolean data structure lower bounds. In *Proc. 50th STOC*, 2017. [doi:10.1145/3188745.3188790, arXiv:1703.03575] 6

[26] OHAD LIPSKY AND ELY PORAT: L_1 pattern matching lower bound. *Inform. Process. Lett.*, 105(4):141–143, 2008. Preliminary version in SPIRE'05. [doi:10.1016/j.ipl.2007.08.011] 3

[27] MARVIN MINSKY AND SEYMOUR PAPERT: *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969. 5

[28] JACQUES MORGENSTERN: Note on a lower bound on the linear complexity of the fast Fourier transform. *J. ACM*, 20(2):305–306, 1973. [doi:10.1145/321752.321761] 4

[29] VICTOR YA. PAN: The trade-off between the additive complexity and the asynchronicity of linear and bilinear algorithms. *Inform. Process. Lett.*, 22(1–2):11–14, 1986. [doi:10.1016/0020-0190(86)90035-9] 4

[30] CHRISTOS H. PAPADIMITRIOU: Optimality of the Fast Fourier Transform. *J. ACM*, 26(1):95–102, 1979. [doi:10.1145/322108.322118] 4

[31] MIKE S. PATERSON, MICHAEL J. FISCHER, AND ALBERT R. MEYER: An improved overlap argument for on-line multiplication. In *SIAM-AMS Proceedings*, volume 7, pp. 97–111. Amer. Math. Soc., 1974. 4, 16, 17

[32] MIHAI PĂTRAŞCU: *Lower bound techniques for data structures*. Ph. D. thesis, MIT, 2008. ACM DL. 5

[33] MIHAI PĂTRAŞCU AND ERIK D. DEMAINE: Tight bounds for the partial-sums problem. In *Proc. 15th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'04)*, pp. 20–29. ACM Press, 2004. ACM DL. 6, 30

[34] MIHAI PĂTRAŞCU AND ERIK D. DEMAINE: Logarithmic lower bounds in the cell-probe model. *SIAM J. Comput.*, 35(4):932–963, 2006. Preliminary versions in [33] and STOC'04. [doi:10.1137/S0097539705447256, arXiv:cs/0502041] 5, 6, 10

[35] DAVID WOODRUFF: Optimal space lower bounds for all frequency moments. In *Proc. 15th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'04)*, pp. 167–175. ACM Press, 2004. ACM DL. 2

[36] ANDREW CHI-CHIH YAO: Probabilistic computations: Toward a unified measure of complexity. In *Proc. 18th FOCS*, pp. 222–227, 1977. Preliminary version in FOCS'78. [doi:10.1109/SFCS.1977.24] 3, 8

[37] ANDREW CHI-CHIH YAO: Should tables be sorted? *J. ACM*, 28(3):615–628, 1981. [doi:10.1145/322261.322274] 5

AUTHORS

Raphaël Clifford
Reader in Algorithm Design
University of Bristol, UK
raphael.clifford@bristol.ac.uk

Markus Jalsenius
Telia Company AB
Solna, Sweden
markus.jalsenius@gmail.com

Benjamin Sach
Alan Turing Institute
London, UK
bsach@turing.ac.uk

ABOUT THE AUTHORS

RAPHAËL CLIFFORD graduated from Oxford University in 1995 and completed his Ph. D. under the supervision of Marek Sergot at Imperial College London. His early work focused on combinatorial pattern matching problem. He now works on streaming algorithms and on conditional and unconditional lower bounds for data structure problems.

MARKUS JALSENIUS completed his Ph. D. in Computer Science at the University of Liverpool in 2008 under the supervision of Leslie Ann Goldberg and has been a Research Associate at both the University of Liverpool and the University of Bristol. Markus is now working as a Data Scientist at Telia Company. The present work was completed while Markus was working at the University of Bristol.

BENJAMIN SACH completed his Ph. D. in Computer Science at the University of Bristol in 2011 under the supervision of Raphaël Clifford. He was an EPSRC fellow at the University of Warwick and subsequently a Lecturer at the University of Bristol. Ben now works as a Data Scientist with the Alan Turing Institute. The present work was completed while Ben was working at the University of Bristol.