Cell-Probe Lower Bounds for Bit Stream Computation

Raphaël Clifford[1], Markus Jalsenius[2], and Benjamin Sach[3]

1 Department of Computer Science, University of Bristol, Bristol, UK
2 Department of Computer Science, University of Bristol, Bristol, UK
3 Department of Computer Science, University of Bristol, Bristol, UK

Abstract

We revisit the complexity of online computation in the cell probe model. We consider a class of problems where we are first given a fixed pattern F of n symbols and then one symbol arrives at a time in a stream. After each symbol has arrived we must output some function of F and the n-length suffix of the arriving stream. Cell probe bounds of $\Omega(\delta \lg n/w)$ have previously been shown for both convolution and Hamming distance in this setting, where δ is the size of a symbol in bits and $w \in \Omega(\lg n)$ is the cell size in bits. However, when δ is a constant, as it is in many natural situations, the existing approaches no longer give us non-trivial bounds.

We introduce a *lop-sided information transfer* proof technique which enables us to prove meaningful lower bounds even for constant size input alphabets. Our new framework is capable of proving amortised cell probe lower bounds of $\Omega(\lg^2 n/(w \cdot \lg\lg n))$ time per arriving *bit*. We demonstrate this technique by showing a new lower bound for a problem known as pattern matching with address errors or the L_2-rearrangement distance problem. This gives the first non-trivial cell probe lower bound for any online problem on bit streams that still holds when the cell size is large.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Cell-probe lower bounds, algorithms, data streaming

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.31

1 Introduction

We revisit the complexity of online computation in the cell probe model. In recent years there has been considerable progress towards the challenging goal of establishing lower bounds for both static and dynamic data structure problems. A third class of data structure problems which fall somewhere between these two classic settings, is online computation in a streaming setting. Here one symbol arrives at a time and a new output must be given after each symbol arrives and before the next symbol is processed. The key conceptual difference to a standard dynamic data structure problem is that although each arriving symbol can be regarded as a new update operation at a prespecified index, there is only one type of query which is to output the latest value of some function of the stream.

Online pattern matching is particularly suited to study in this setting and cell probe lower bounds have previously been shown for different measures of distance including Hamming distance, inner product/convolution and edit distance [3, 4, 5]. All these previous cell probe lower bounds have relied on only one proof technique, the so-called *information transfer technique* of Pătraşcu and Demaine [14]. In loose terms the basic idea is as follows. First one defines a random input distribution over updates. Here we regard an arriving symbol as an update and after each update we perform one query which simply returns the latest

24th Annual European Symposium on Algorithms (ESA 2016).
Editors: Piotr Sankowski and Christos Zaroliagis; Article No. 31; pp. 31:1–31:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

distance between a predefined pattern and the updated suffix of the stream. One then has to argue that knowledge of the answers to ℓ consecutive queries is sufficient to infer at least a constant fraction of the information encoded by ℓ consecutive updates that occurred in the past. If one can show this is true for all power of two lengths ℓ then a logarithmic lower bound per update/query operation follows.

In recent years a consensus has been arrived at that the most natural cell size is $w \in \Omega(\lg n)$. This is for two main reasons. The first is simply that a cell should be large enough to be able to address all of memory. The second, more practical reason is that lower bounds that we derive directly give time lower bounds for problems analysed in the popular word-RAM model. When cells are of this size a cell probe lower bound of $\Omega(\delta \lg n/w)$ for both online Hamming distance and convolution using the information transfer technique has been shown, where δ is the number of bits in an input symbol, w is the cell size in bits and n is the length of the fixed pattern [3, 4]. When $\delta \geq w \geq \lg n$, there is also a matching upper bound in the cell probe model and so no further progress is possible. However, when the symbol size δ does not grow with the input size as is often the case in applied settings, the best lower bound that is derivable reduces trivially to be constant. This is an unfortunate situation as a particularly natural setting of parameters is when the input alphabet is of constant size but the cell size is not.

This small input alphabet, large cell size setting has received some study in the past. Using a sophisticated variant of the information transfer technique, Pătraşcu and Demaine [14] proved an $\Omega(\lg n/ \lg\lg n)$ cell probe lower bound for the classic prefix sum problem when the random update *values* contain $\delta = O(1)$ bits and the cell size is $\Theta(\lg n)$. However, as they themselves highlight in their paper, their proof technique relies on the fact that the update *indices* contain $\Omega(\lg n)$ random bits and it is this information which is then used to provide the lower bound. This is in contrast to our streaming setting where both the update and query indices are fixed and the update values contain only a constant number of bits each.

In this paper we introduce a new variant of the information transfer technique which we call the *lop-sided information transfer technique*. This will enable us to give meaningful lower bounds for precisely this setting, that is when $\delta \in O(1)$, $w \in \Omega(\lg n)$ and both the query and update indices are fixed. Our proof technique will rely on being able to show for specific problems that we need only ℓ query answers to infer at least a constant fraction of the information encoded in the previous $\ell \lg \ell$ updates.

We demonstrate our new framework by applying it to a pattern matching problem with address errors known as L_2-rearrangement distance. This measure of distance, which was first studied in SODA 2006 [1, 2], arises in pattern matching problems where errors occur not in the content of the data but in the addresses where the data is stored. Our proof technique is fundamentally combinatorial in nature. We demonstrate an input distribution which has the property that individual bits of the $\Theta(\lg n)$ sized outputs encode individual bits of the stream. In this way we can infer $\Omega(\ell \lg \ell)$ updates from only $O(\ell)$ outputs as we require. We believe our proof technique is also directly applicable to other simpler distance measures such as the Hamming distance. However establishing the key technical lemma (Lemma 5) appears to be out of reach at present.

The cell probe model and previous lower bounds

Our bounds hold in a particularly strong computational model, the *cell-probe model*, introduced originally by Minsky and Papert [11] in a different context and then subsequently by Fredman [7] and Yao [17]. In this model, there is a separation between the computing unit and the memory, which is external and consists of a set of cells of w bits each. The computing unit cannot remember any information between operations. Computation is free and the cost is measured only in the number of cell reads or writes (cell-probes). This general view makes the model very strong, subsuming for instance the popular word-RAM model.

The first techniques known for establishing dynamic data structure lower bounds had historically been based on the chronogram technique of Fredman and Saks [8] which can at best give bounds of $\Omega(\lg n/\lg\lg n)$. In 2004, Pătraşcu and Demaine gave us the first $\Omega(\lg n)$ lower bounds for dynamic data structure problems [14]. Their technique is based on information theoretic arguments which also form the basis for the work we present in this paper. 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 [12]. More recently, a further breakthrough was made by Larsen who showed lower bounds of roughly $\Omega((\lg n/\lg\lg n)^2)$ time per operation for dynamic weighted range counting problem and polynomial evaluation [9, 10]. Subsequent application of this new proof technique has also provided the same lower bound for dynamic matrix-vector multiplication [15]. These lower bounds remain the state of the art for any dynamic structure problem to this day. It is particularly relevant that Larsen's lower bound for dynamic weighted range counting problem cannot yet be applied to the unweighted range counting problem due to a very similar limitation in proof technique to the one we address in this paper.

1.1 Our Results

The lop-sided information transfer technique

In the standard formulation of Demaine and Pătraşcu's information transfer technique [13], two adjacent time intervals $[t_0, t_1]$ and $[t_1+1, t_2]$ are considered, with equal, power of two length ℓ. To apply this technique in a streaming setting, the core argument one has to make is that for the given problem, knowledge of the outputs during $[t_1+1, t_2]$ is sufficient to infer a constant fraction of the information encoded by updates during $[t_0, t_1]$. As this information about the updates can be inferred *from* the outputs, the algorithm must know this information *to compute* the outputs. In particular this implies that while computing the outputs during $[t_1+1, t_2]$, the algorithm must probe sufficiently many cells written during $[t_0, t_1]$ to uniquely recover this information. We can think of these cell probes as being associated with interval length ℓ and offset t_0 which uniquely defines the two intervals. The final lower bound is obtained by summing the cell probe lower bounds associated with every power-of-two length ℓ and $t_0 = \ell, 2\ell, 3\ell \ldots$. This final step relies crucially on the fundamental property of the information transfer technique that this summation step does not double count cell probes. In particular that a cell probe associated with some t_0, ℓ is not also associated with some other t_0', ℓ'.

As the argument is information theoretic, to obtain a *logarithmic* lower bound via this approach, both the ℓ updates during $[t_0, t_1]$ and the ℓ outputs during $[t_1+1, t_2]$ must contain $\Omega(\ell\lg\ell)$ bits. However in the bit streaming setting, each update contains $O(1)$ bits so the updates in $[t_0, t_1]$ contain only $O(\ell)$ bits in total.

To overcome this we increase the size of the interval $[t_0, t_1]$ to have length $\ell\lg\ell$ so that both intervals contain $\Omega(\ell\lg\ell)$ bits as required. Unfortunately this modification breaks the fundamental property of the information transfer technique that there is no double counting of cell probes. In fact, direct application of our approach causes each cell probe to be counted $\Theta(\lg n)$ times, negating the possibility of a non-trivial lower bound.

To overcome this we place gaps in time between the end of one interval and the start of another and argue carefully both that not too much of the information can be lost in these gaps and that we can still sum the cell probes over a sufficient number of distinct interval lengths without too much double counting. Our hope is that this new technique will lead to a new class of cell probe lower bounds which could not be proved with existing methods.

Online pattern matching with address errors (L_2-rearrangement distance)

We give an explicit distance function for which we can now obtain the first unconditional online cell probe lower bound for symbol size $\delta = 1$. Consider two strings S_1 and S_2 both of length n where S_2 is a permutation of S_1. Now consider the set of permutations Π so that for all $\pi \in \Pi$, $S_1[\pi(0), \ldots, \pi(n-1)] = S_2$. The L_2-rearrangement distance is defined to be $\min_{\pi\in\Pi} \sum_{j=0}^{n-1}(j - \pi(j))^2$ [2]. In other words, the cost of a permutation is the sum of the square of the number of positions each character is moved. The distance is the minimum cost of any permutation. If Π is empty, that is S_2 is in fact not a permutation of S_1, then the L_2-rearrangement distance is defined to be ∞. As an example, the L_2-rearrangement distance between strings 11100 and 10110 is $0{+}1{+}1{+}2^2{+}0{=}6$. In the online L_2-rearrangement problem we are given a fixed pattern $F \in \{0,1\}^n$ and the stream arrives one symbol at a time. After each symbol arrives we must output the L_2-rearrangement distance between F and the most recent n-length suffix of the stream. This online version can be solved in $O(\lg^2 n)$ time per arriving symbol in the word-RAM model [6].

Our technique allows us to recover $\Omega(\lg n)$ distinct bits of the stream from each output. This is achieved by constructing F and carefully choosing a highly structured random input distribution for the incoming stream in such a way that the contributions to the output from different regions of the stream have different magnitudes. We can then use the result to extract distinct information about the stream from different parts of each output.

Using this approach we get the following cell probe lower bound:

▶ **Theorem 1** (Online L_2-rearrangement). *In the cell-probe model with $w \in \Omega(\lg n)$ bits per cell, for any randomised algorithm solving the online L_2-rearrangement distance problem on binary inputs there exist instances such that the expected amortised number of probes per arriving value is*

$$\Omega\left(\frac{\lg^2 n}{w \cdot \lg\lg n}\right).$$

2 Lop-sided information transfer

In this section we will formally define our variant of information transfer, which is a particular set of cells probed by the algorithm, and explain how a bound on the size of the information transfer can be used when proving the overall lower bound of Theorems 1. Our lower bound holds for any randomised algorithm on its worst case input. This will be achieved by applying *Yao's minimax principle* [16]. As a result, from this point onwards we consider an arbitrary deterministic algorithm running with some fixed array F on a random input of n stream values over the binary alphabet $\Sigma = \{0, 1\}$. The algorithm may depend on F. As is common in the literature we will refer to the choice of F and the distribution of stream values as the *hard distribution.*

We will let $U \in \{0,1\}^n$ denote the *update array* which describes a sequence of n update operations corresponding to values that arrive in the stream. We will usually refer to the t-th update as the *arrival* of the value $U[t]$. Observe that just after arrival t, the values

$U[t+1, n-1]$ are still not known to the algorithm. We will proceed under the assumption that before the 0-th update, $U[0]$ arrives, the stream contains at least n symbols chosen arbitrarily from the support of the stream distribution. All logarithms are in base two.

2.1 Notation – Two intervals and a gap

In order to define the concept of information transfer from one interval of arriving values in the stream to another interval of arriving values, we first define the set L which contains the interval lengths that we will consider,

$$L = \left\{ n^{1/4} \cdot (\lg n)^{2i} \;\middle|\; i \in \left\{0, 1, 2, \ldots, \frac{\lg n}{4 \lg \lg n}\right\} \right\}.$$

To avoid cluttering the presentation with floors and ceilings, we assume throughout that the value of n is such that any division or power nicely yields an integer. Whenever it is impossible to obtain an integer we assume that suitable floors or ceilings are used. In particular, L contains only integers.

In contrast to the original information transfer method, we define *three* intervals $[t_0, t_1]$, $[t_1+1, t_2-1]$ and $[t_2, t_3]$, referred to as the *left interval*, the *gap* and the *right interval*, respectively. These intervals are functions of a length $\ell \in L$ and an offset $t \in [n/2]$. The left interval has length $\ell \lg \ell$, the gap has length $4\ell/\lg n$ and the right interval has length ℓ. Precisely we define the following four values:

$$t_0 = t, \qquad t_1 = t_0 + \ell \lg \ell - 1, \qquad t_2 = t_1 + \frac{4\ell}{\lg n} + 1, \qquad t_3 = t_2 + \ell - 1.$$

Formally the values t_0, t_1, t_2 and t_3 are functions of ℓ and t but for brevity we will often write just t_0 instead of $t_0(\ell, t)$, and so on, whenever the parameters ℓ and t are obvious from context.

We now highlight some useful properties of these intervals which are easily verified. First observe that the intervals are disjoint and that all intervals are contained in $[0, n-1]$ for sufficiently large n. Second, suppose that $\ell' \in L$ is one size larger than $\ell \in L$, that is $\ell' = \ell \cdot (\lg n)^2$. For ℓ' the length of the gap is $4\ell'/\lg n$, which is sufficiently large that it spans the length of the left interval, the right interval and the gap associated with ℓ. This second property will be particularly important in proving that we do not over-count cell probes.

2.2 Information transfer over gaps

Towards the definition of information transfer, we define, for $\ell \in L$ and $t \in [n/2]$, the subarray $U_{\ell,t} = U[t_0, \ldots, t_1]$ to represent the $\ell \lg \ell$ values arriving in the stream during the left interval. We define the subarray $A_{\ell,t}$ to represent the ℓ outputs during the right interval $[t_2, \ldots, t_3]$. Lastly we define $\widetilde{U}_{\ell,t}$ to be the concatenation of $U[0, (t_0-1)]$ and $U[(t_1+1), (n-1)]$. That is, $\widetilde{U}_{\ell,t}$ contains all values of U except for those in $U_{\ell,t}$.

For $\ell \in L$ and $t \in [n/2]$ we first define the *information transfer to the gap*, denoted $\mathcal{G}_{\ell,t}$, to be the set of memory cells c such that c is probed during the left interval $[t_0, t_1]$ of arriving values and also probed during the arrivals of the values $U[t_1+1, t_2-1]$ in the gap. Similarly we define the information transfer to the right interval, or simply *the information transfer*, denoted $\mathcal{I}_{\ell,t}$, to be the set of memory cells c such that c is probed during the left interval $[t_0, t_1]$ of arriving symbols and also probed during the arrivals of symbols in the right interval $[t_2, t_3]$ but *not* in the gap. That is, any cell $c \in \mathcal{G}_{\ell,t}$ cannot also be contained in the information transfer $\mathcal{I}_{\ell,t}$.

The cells in the information transfer $\mathcal{I}_{\ell,t}$ may contain information about the values in $U_{\ell,t}$ that the algorithm uses in order to correctly produce the outputs $A_{\ell,t}$. However, since cells that are probed in the gap are not included in the information transfer, the information transfer might not contain all the information about the values in $U_{\ell,t}$ that the algorithm uses while outputting $A_{\ell,t}$. We will see that the gap is small enough that a large fraction of the information about $U_{\ell,t}$ has to be fetched from cells in the information transfer $\mathcal{I}_{\ell,t}$.

Since cells in the information transfer are by definition probed at some point by the algorithm, we can use $\mathcal{I}_{\ell,t}$ to measure, or at least give a lower bound for, the number of cell probes. As a shorthand we let $I_{\ell,t} = |\mathcal{I}_{\ell,t}|$ denote the size of the information transfer $\mathcal{I}_{\ell,t}$. Similarly we let $G_{\ell,t} = |\mathcal{G}_{\ell,t}|$ denote the size of the information transfer to the gap. By adding up the sizes $I_{\ell,t}$ of the information transfers over all $\ell \in L$ and certain values of $t \in [n/2]$, we get a lower bound on the total number of cells probed by the algorithm during the n arriving values in U. The choice of the values t is crucial as we do not want to over-count the number of cell probes. In the next two lemmas we will deal with the potential danger of over-counting.

For a cell $c \in \mathcal{I}_{\ell,t}$, we write *the probe of c with respect to $\mathcal{I}_{\ell,t}$* to refer to the first probe of c during the arrivals in the right interval. These are the probes of the cells in the information transfer that we count.

▶ **Lemma 2.** *For any $\ell \in L$ and $t, t' \in [n/2]$ such that $|t - t'| \geq \ell$, if a cell c is in both $\mathcal{I}_{\ell,t}$ and $\mathcal{I}_{\ell,t'}$ then the probe of c with respect to $\mathcal{I}_{\ell,t}$ and the probe of c with respect to $\mathcal{I}_{\ell',t'}$ are distinct.*

Proof. Since t and t' are at least ℓ apart, the right intervals associated with t and t', respectively, must be disjoint. Hence the probe of c with respect to $\mathcal{I}_{\ell,t}$ and the probe of c with respect $\mathcal{I}_{\ell,t'}$ must be distinct. ◀

From the previous lemma we know that there is no risk of over-counting cell probes of a cell over information transfers $\mathcal{I}_{\ell,t}$ under a fixed value of $\ell \in L$, as long as no two values of t are closer than ℓ. The proof follows directly from the fact that as $|t - t'| \geq \ell$, the corresponding right intervals for t and t' do not overlap. Distinctness then follows directly from the definition of information transfer. In the next lemma we consider information transfers under different values of $\ell \in L$. The proof follows from the property introduced in Section 2.1 that if (wlog.) $\ell' > \ell$, the gap associated with ℓ' is spans all three intervals associated with ℓ. This implies that either the right intervals for t and t' do not overlap or the left intervals do not overlap. In either case, once again, distinctness follows directly from the definition of information transfer.

▶ **Lemma 3.** *For any $\ell, \ell' \in L$ such that $\ell \neq \ell'$, and any $t, t' \in [n/2]$, if a cell c is in both $\mathcal{I}_{\ell,t}$ and $\mathcal{I}_{\ell',t'}$ then the probe of c with respect to $\mathcal{I}_{\ell,t}$ and the probe of c with respect to $\mathcal{I}_{\ell',t'}$ must be distinct.*

Proof. Let p be the probe of c with respect to $\mathcal{I}_{\ell,t}$, and let p' be the probe of c with respect $\mathcal{I}_{\ell',t'}$. We will show that $p \neq p'$. Suppose without loss of generality that $\ell < \ell'$. From the properties of the intervals that were given in the previous section we know that the length of the gap associated with ℓ' is larger than the sum of lengths of the left interval, the gap and the right interval associated with ℓ.

Suppose for contradiction that $p = p'$. By definition of $\mathcal{I}_{\ell,t}$, the cell c is probed also in the left interval associated with ℓ. Let p_{first} denote any such cell probe. Because the gap associated with ℓ' is so large, p_{first} must take place either in the right interval or the gap associated with ℓ'. If p_{first} is in the gap, then c cannot be in $\mathcal{I}_{\ell',t'}$. If p_{first} is in the right interval then p' cannot equal p. ◀

In order to give a lower bound for the total number of cell probes performed by the algorithm over the n arrivals in U we will define, for each $\ell \in L$, a set $T_\ell \subseteq [n/2]$ of arrivals, such that for any distinct $t, t' \in T_\ell$, $|t - t'| \geq \ell$. It then follows from Lemmas 2 and 3 that

$$\sum_{\ell \in L} \sum_{t \in T_\ell} I_{\ell,t}$$

is a lower bound on the number of cell probes. Our goal is to give a lower bound for the expected value of this double-sum. The exact definition of T_ℓ will be given in Section 3.3 once we have introduced relevant notation.

3 Proving the lower bound

In this section we give the overall proof for the lower bound of Theorem 1. Let $\ell \in L$ and let $t \in [n/2]$. Suppose that $\widetilde{U}_{\ell,t}$ is fixed but the values in $U_{\ell,t}$ are drawn at random in accordance with the distribution for U, conditioned on the fixed value of $\widetilde{U}_{\ell,t}$. This induces a distribution for the outputs $A_{\ell,t}$. We want to show that if the entropy of $A_{\ell,t}$ is large, conditioned on the fixed $\widetilde{U}_{\ell,t}$, then the information transfer $\mathcal{I}_{\ell,t}$ is large, since only the variation in the inputs $U_{\ell,t}$ can alter the outputs $A_{\ell,t}$. We will soon make this claim more precise.

3.1 Upper bound on entropy

We write $H(A_{\ell,t} \mid \widetilde{U}_{\ell,t} = \widetilde{u}_{\ell,t})$ to denote the entropy of $A_{\ell,t}$ conditioned on fixed $\widetilde{U}_{\ell,t}$. Towards showing that high conditional entropy $H(A_{\ell,t} \mid \widetilde{U}_{\ell,t} = \widetilde{u}_{\ell,t})$ implies large information transfer we use the information transfer $\mathcal{I}_{\ell,t}$ and the information transfer to the gap, $\mathcal{G}_{\ell,t}$, to describe an encoding of the outputs $A_{\ell,t}$. The following lemma gives a direct relationship between $I_{\ell,t} + G_{\ell,t}$ and the entropy which is applicable to both of our online problems. A marginally simpler version of the lemma, stated with different notation, was first given in [14] under the absence of gaps.

▶ **Lemma 4.** *Under the assumption that the address of any cell can be specified in w bits, for any $\ell \in L$ and $t \in [n/2]$, the entropy $H(A_{\ell,t} \mid \widetilde{U}_{\ell,t} = \widetilde{u}_{\ell,t}) \leq 2w + 2w \cdot \mathbb{E}[I_{\ell,t} + G_{\ell,t} \mid \widetilde{U}_{\ell,t} = \widetilde{u}_{\ell,t}]$.*

Proof. The expected length of any encoding of $A_{\ell,t}$ under fixed $\widetilde{U}_{\ell,t}$ is an upper bound on the conditional entropy of $A_{\ell,t}$. We use the information transfer $\mathcal{I}_{\ell,t}$ and the information transfer to the gap, $\mathcal{G}_{\ell,t}$, to define an encoding of $A_{\ell,t}$ in the following way. For every cell $c \in \mathcal{I}_{\ell,t} \cup \mathcal{G}_{\ell,t}$ 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 that it holds at the very end of the left interval, just before the beginning of the gap. The contents of c is specified with w bits. In total this requires $2w \cdot (I_{\ell,t} + G_{\ell,t})$ bits.

We will use the algorithm, which is fixed, and the fixed values $\widetilde{u}_{\ell,t}$ of $\widetilde{U}_{\ell,t}$ as part of the decoder to obtain $A_{\ell,t}$ from the encoding. Since the encoding is of variable length we also store the size $I_{\ell,t}$ of the information transfer and the size $G_{\ell,t}$ of the information transfer to the gap. This requires at most $2w$ additional bits.

In order to prove that the described encoding of $A_{\ell,t}$ is valid we now describe how to decode it. First we simulate the algorithm on the fixed input $\widetilde{U}_{\ell,t}$ from the first arrival $U[0]$ until just before the left interval when the first value in $U_{\ell,t}$ arrives. We then skip over all inputs in $U_{\ell,t}$ and resume simulating the algorithm from the beginning of the gap, that is when the value $U[t_1+1]$ arrives. We simulate the algorithm over the arrivals in the gap and the right interval until all values in $A_{\ell,t}$ have been outputted. For every cell being read, we check if it is contained in either the information transfer $\mathcal{I}_{\ell,t}$ or the information transfer to the gap $\mathcal{G}_{\ell,t}$ by looking up its address in the encoding. If the address is found then the contents of the cell is fetched from the encoding. If not, its contents is available from simulating the algorithm on the fixed inputs $\widetilde{U}_{\ell,t}$. ◀

3.2 Lower bound on entropy

Lemma 4 above provides a direct way to obtain a lower bound on the expected value of $I_{\ell,t}+G_{\ell,t}$ if given a lower bound on the conditional entropy $H(A_{\ell,t} \mid \widetilde{U}_{\ell,t}=\widetilde{u}_{\ell,t})$. In Lemma 5 we provide such an entropy lower bound for L_2-rearrangement distance. The proof is deferred to Section 4.

▶ **Lemma 5.** *For the L_2-rearrangement distance problem there exists a real constant $\kappa>0$ and, for any n, a fixed array $F\in\{0,1\}^n$ such that for all $\ell\in L$ and all $t\in[n/2]$ such that $t \bmod 4 = 0$, when U is chosen uniformly at random from $\{0101,1010\}^{\frac{n}{4}}$ then,*

$$H(A_{\ell,t} \mid \widetilde{U}_{\ell,t}=\widetilde{u}_{\ell,t}) \geq \kappa\cdot\ell\cdot\lg n, \text{ for any fixed } \widetilde{u}_{\ell,t}.$$

Before we proceed with the lower bound on the information transfer we make a short remark on the bounds that this lemmas gives. Observe that the maximum conditional entropy of $A_{\ell,t}$ is bounded by the entropy of $U_{\ell,t}$, which is $O(\ell\lg\ell)$ since the length of the left interval is $\ell\lg\ell$. Recall also that the values in L range from $n^{1/4}$ to $n^{3/4}$. Thus, for a constant κ, the entropy lower bound is tight up to a multiplicative constant factor.

3.3 A lower bound on the information transfer and quick gaps

In this section we prove our main lower bound results. We assume that κ is the constant and F is the fixed array of Lemma 5, and that U is chosen uniformly at random from $\{0101,1010\}^{\frac{n}{4}}$.

By combining the upper and lower bounds on the conditional entropy from Lemmas 4 and 5 we have that there is a hard distribution and a real constant $\kappa>0$ such that,

$$\mathbb{E}[I_{\ell,t}+G_{\ell,t} \mid \widetilde{U}_{\ell,t}=\widetilde{u}_{\ell,t}] \geq \frac{\kappa\cdot\ell\cdot\lg n}{2w}-1 \text{ for any } \widetilde{u}_{\ell,t}.$$

We may remove the conditioning by taking expectation over $\widetilde{U}_{\ell,t}$ under random U. Thus,

$$\mathbb{E}[I_{\ell,t}+G_{\ell,t}] \geq \frac{\kappa\cdot\ell\cdot\lg n}{2w}-1, \text{ or equivalently,}$$

$$\mathbb{E}[I_{\ell,t}] \geq \frac{\kappa\cdot\ell\cdot\lg n}{2w}-1-\mathbb{E}[G_{\ell,t}]. \tag{1}$$

Recall that our goal is to give a lower bound for

$$\mathbb{E}\left[\sum_{\ell\in L}\sum_{t\in T_\ell} I_{\ell,t}\right]=\sum_{\ell\in L}\sum_{t\in T_\ell}\mathbb{E}\left[I_{\ell,t}\right], \text{ where } T_\ell \text{ contains suitable values of } t.$$

Using inequality (1) would immediately provide such a lower bound, however, there is an imminent risk that the $\mathbb{E}[G_{\ell,t}]$ terms could devalue such a bound into something trivially small. Now, for this to happen, the algorithm must perform sufficiently many cell probes in the gap. Since the length of the gap is considerably shorter than the right interval, a cap on the worst-case number of cell probes per arriving value would certainly ensure that $\mathbb{E}[G_{\ell,t}]$ stays small, but as we want a stronger amortised lower bound we need something more refined. The answer lies in how we define T_ℓ. We discuss this next.

For $\ell \in L$ and $f \in [\ell]$ we first define $T_{\ell,f} = \left\{ f + i\ell \,\middle|\, i \in \{0,1,2,\dots\} \text{ and } f + i\ell \le \frac{n}{2} \right\}$ to be the set of arrivals. The values in $T_{\ell,f}$ are evenly spread out, distance ℓ apart, starting at f. We may think of f as the offset of the sequence of values in $T_{\ell,f}$. The largest value in the set is no more than $n/2$. We will define the set T_ℓ to equal a subset of one of the sets $T_{\ell,f}$ for some f. More precisely, we will show that there must exist an offset f such that at least half of the values $t \in T_{\ell,f}$ have the property that the time spent in the gap associated with ℓ and t is small enough to ensure that the information transfer to the gap is small. We begin with some definitions.

▶ **Definition 6** (Quick gaps and sets)**.** For any $\ell \in L$ and $t \in [n/2]$ we say that the gap associated with ℓ and t is *quick* if the expected number of cell probes during the arrivals in the gap is no more than $\kappa\ell \lg n/(4w)$, where κ is the constant from Lemma 5. Further, for any $f \in [\ell]$ we say that the set $T_{\ell,f}$ is *quick* if, for at least half of all $t \in T_{\ell,f}$, the gap associated with ℓ and t is quick.

The next lemma says that for sufficiently fast algorithms there is always an offset f such that $T_{\ell,f}$ is quick. The proof intuition is that if $T_{\ell,f}$ is not quick for any offset f then the whole algorithm must be slow which gives a contradiction.

▶ **Lemma 7.** *Suppose that the expected total number of cell probes over the n arrivals in U is less than $\kappa n(\lg^2 n)/(32w)$. Then, for any $\ell \in L$, there is an $f \in [\ell]$ such that $T_{\ell,f}$ is quick.*

Proof. In accordance with the lemma, suppose that the expected total number of cell probes over the n arrivals in U is less than $\kappa n(\lg^2 n)/(32w)$. For contradiction, suppose that there is no $f \in [\ell]$ such that $T_{\ell,f}$ is quick. We will show that the expected number of cell probes over the n arrivals must then be at least $\kappa n(\lg^2 n)/(32w)$.

For any $f \in [\ell]$, let $R_f \subseteq [n]$ be the union of all arrivals that belong to a gap associated with ℓ and any $t \in T_{\ell,f}$. Let P_f be the number of cell probes performed by the algorithm over the arrivals in R_f. Thus, for any set $T_{\ell,f}$ that is *not quick* we have by linearity of expectation $\mathbb{E}[P_f] \ge \frac{|T_{\ell,f}|}{2} \cdot \frac{\kappa \cdot \ell \cdot \lg n}{4w} = \frac{n/2}{2\ell} \cdot \frac{\kappa\cdot\ell\cdot\lg n}{4w} = \frac{\kappa\cdot n\cdot\lg n}{8w}$.

Let the set of offsets $\mathcal{F} = \left\{ i \cdot \frac{4\ell}{\lg n} \,\middle|\, i \in \left[\frac{\lg n}{4}\right] \right\} \subseteq [\ell]$. The values in $\mathcal{F}$ are spread out with distance $4\ell/\lg n$, which equals the gap length. Thus, for any two distinct $f, f' \in \mathcal{F}$, the sets R_f and $R_{f'}$ are disjoint. We therefore have that the total running time over all n arrivals in U must be bounded below by $\sum_{f\in\mathcal{F}} P_f$. Under the assumption that no $T_{\ell,f}$ is quick, we have that the expected total running time is at least $\mathbb{E}\left[\sum_{f\in\mathcal{F}} P_f\right] = \sum_{f\in\mathcal{F}} \mathbb{E}[P_f] \ge |\mathcal{F}| \cdot \frac{\kappa\cdot n\cdot\lg n}{8w} = \frac{\lg n}{4} \cdot \frac{\kappa\cdot n\cdot\lg n}{8w} = \frac{\kappa\cdot n\cdot\lg^2 n}{32w}$, which is the contradiction we wanted. Thus, under the assumption that the running time over the n arrivals in U is less than $\kappa n(\lg^2 n)/(32w)$ there must be an $f \in [\ell]$ such that $T_{\ell,f}$ is quick. ◀

We now proceed under the assumption that the expected running time over the n arrivals in U is less than $\kappa n(\lg^2 n)/(32w)$. If this is not the case then we have already established the lower bound of Theorem 1.

Let f be a value in $[\ell]$ such that $T_{\ell,f}$ is a quick set. Such an f exists due to Lemma 7. We now let $T_\ell \subseteq T_{\ell,f}$ be the set of all $t \in T_{\ell,f}$ for which the gap associated with ℓ and t is quick. Hence $|T_\ell| \geq |T_{\ell,f}|/2 = n/(4\ell)$. Since $G_{\ell,t}$ cannot be larger than the number of cell probes in the gap, we have by the definition of a quick gap that for any $t \in T_\ell$,

$$\mathbb{E}\left[G_{\ell,t}\right] \leq \frac{\kappa \cdot \ell \cdot \lg n}{4w}.$$

By combining the inequalities we can finally provide a non-trivial lower bound on the sum of the information transfers:

$$\begin{aligned}\sum_{\ell \in L, t \in T_\ell} \mathbb{E}\left[I_{\ell,t}\right] &\geq \sum_{\ell \in L, t \in T_\ell} \left(\frac{\kappa \cdot \ell \cdot \lg n}{2w} - 1 - \mathbb{E}[G_{\ell,t}]\right) \\ &\geq \sum_{\ell \in L, t \in T_\ell} \left(\frac{\kappa \cdot \ell \cdot \lg n}{2w} - 1 - \frac{\kappa \cdot \ell \cdot \lg n}{4w}\right) \geq \frac{\kappa \cdot \lg n}{5w} \sum_{\ell \in L, t \in T_\ell} \ell \\ &\geq \frac{\kappa \cdot \lg n}{5w} \sum_{\ell \in L} (|T_\ell| \cdot \ell) \geq \frac{\kappa \cdot \lg n}{5w} \sum_{\ell \in L} \left(\frac{n}{4\ell} \cdot \ell\right) = \frac{\kappa \cdot n \cdot \lg n}{20w} \cdot |L| \\ &\geq \frac{\kappa \cdot n \cdot \lg n}{20w} \cdot \frac{\lg n}{4 \lg \lg n} \in \Theta\left(\frac{n \cdot \lg^2 n}{w \cdot \lg \lg n}\right).\end{aligned}$$

By Lemmas 2 and 3 this lower bound is also a bound on the expected total number of cell probes performed by the algorithm over the n arrivals in U. The amortised time per arriving value is obtained by dividing the running time by n, concluding the proof of Theorem 1.

4 The hard distribution for L_2-rearrangement

In this section we prove Lemma 5. Recall that U is chosen uniformly at random from $\{0101, 1010\}^{\frac{n}{4}}$. For each $\ell \in L$ there is a subarray of F of length $\ell \lg \ell + \ell$. Each such subarray, which we denote F_ℓ, is at distance $4\ell/\lg n + 1$ from the right-hand end of F, which is one more than the length of the gap associated with ℓ. By the properties discussed in Section 2.1 we know that the length of the gap associated with ℓ' is larger than the length of F_ℓ plus the length of the gap associated with ℓ. Hence there is no overlap between the subarrays F_ℓ and $F_{\ell'}$.

Given any of the subarrays F_ℓ and an array U_ℓ of length $(\ell \lg \ell)$, we write $F_\ell \odot U_\ell$ to denote the $(\ell/4)$-length array that consists of the L_2-rearrangement distances between U_ℓ and every *fourth* $(\ell \lg \ell)$-length substring of F_ℓ. More precisely, for $4i \in [\ell]$, the value of $F_\ell \odot U_\ell[i]$ is the L_2-rearrangement distance between $F_\ell[4i, 4i + \ell \lg \ell - 1]$ and U_ℓ.

The main focus of this section is proving Lemma 8 which can be seen as an analogue of Lemma 5 for a fixed length of ℓ:

▶ **Lemma 8.** *There exists a real constant $\epsilon > 0$ such that for all n and $\ell \in L$ there is a subarray F_ℓ for which the entropy of $F_\ell \odot U_\ell$ is at least $\epsilon \cdot \ell \lg \ell$ when U_ℓ is drawn uniformly at random from $\{0101, 1010\}^{\frac{\ell}{4} \lg \ell}$. F_ℓ contains an equal number of 0s and 1s.*

In order to finish the description of the array F we choose each subarray F_ℓ in accordance with Lemma 8. Any region of F that is not part of any of the subarrays F_ℓ is filled with repeats of '01'. This ensures that these regions contain an equal number of zeros and ones. This concludes the description of the array F.

The proof of Lemma 5 then follows from Lemma 8 by arguing that the outputs in $F_\ell \odot U_\ell$ can be calculated from the outputs in $A_{\ell,t}$ by subtracting the contributions from $F_{\ell'} \odot U_{\ell'}$ for all $\ell' \neq \ell$. As each required value from $U_{\ell'}$ is contained in $\widetilde{U}_{\ell,t}$ which is fixed to equal $\widetilde{u}_{\ell,t}$, we have that $H(A_{\ell,t} \mid \widetilde{U}_{\ell,t} = \widetilde{u}_{\ell,t}) \geq H(F_\ell \odot U_\ell)$ as required. This argument requires that for each output, the globally optimal (lowest cost) permutation is always compatible with the locally optimal permutation of each U_ℓ. In particular we need to rule out the possibility of characters from some U_ℓ being moved to positions in $F_{\ell'}$ for $\ell \neq \ell'$. The proof (and the lower bound in general) relies on a key property of L_2-arrangement (proven in Lemma 3.1 from [1]) which states that under the optimal permutation, the i-th one (resp. zero) in one string is moved to the i-th one (resp. zero) in the other. By controlling how the zeros and ones are distributed in U and F, we can limit how far any character is moved. For brevity the details are left for the full version.

We are now ready to prove Lemma 5, the lower bound on the conditional entropy of $A_{\ell,t}$.

Proof of Lemma 5. Let F be the array described above and let U be drawn uniformly at random from $\{0101, 1010\}^{\frac{n}{4}}$. Let $\ell \in L$ and $t \in [n/2]$. Thus, conditioned on any fixed $\widetilde{U}_{\ell,t}$, the distribution of $U_{\ell,t}$ is uniform on $\{0101, 1010\}^{\frac{\ell}{4}\lg \ell}$.

Recall that $U_{\ell,t}$ arrives in the stream between arrival t_0 and t_1, after which $4\ell/\lg n$ values arrive in the gap. Thus, at the beginning of the right interval, at arrival t_2, $U_{\ell,t}$ is aligned with the $(\ell \lg \ell)$-length suffix of the subarray F_ℓ of F. Over the ℓ arrivals in the right interval, $U_{\ell,t}$ slides along F_ℓ. We now prove that since all values in $\widetilde{U}_{\ell,t}$ are fixed, the outputs $A_{\ell,t}$ uniquely specify $F_\ell \odot U_{\ell,t}$. The analogous property for convolution was immediate. First observe that by construction the prefix of F up to the start of F_ℓ contains an equal number of 0s and 1s. Similarly for F_ℓ itself and the suffix from F_ℓ to the end of F. Once in every four arrivals, the substring of U aligned with F is guaranteed (by construction) to also have an equal number of 0s and 1s. Therefore the L_2-rearrangement distance is finite. It was proven in Lemma 3.1 from [1] that (rephrased in our notation) under the optimal rearrangement permutation, the k-th one (resp. zero) in F is moved to the k-th one (resp. zero) in U. Therefore, every element of U_ℓ is moved to an element in F_ℓ. We can therefore recover any output in $F_\ell \odot U_{\ell,t}$ by taking the corresponding output in $A_{\ell,t}$ and subtracting, the costs of moving the elements that are in U but not in U_ℓ. It is easily verified that as t is divisible by four, the corresponding output in $A_{\ell,t}$ is one of those guaranteed to have an equal number of 0s and 1s. Thus, by Lemma 8, the conditional entropy

$$H(A_{\ell,t} \mid \widetilde{U}_{\ell,t} = \widetilde{u}_{\ell,t}) \geq \epsilon \cdot \ell \cdot \lg \ell, \geq \frac{\epsilon}{4} \cdot \ell \cdot \lg n,$$

since $\ell \geq n^{1/4}$. By setting the constant κ to $\epsilon/4$ we have proved Lemma 5. ◀

4.1 High entropy for fixed ℓ – the proof of Lemma 8

In this section we prove Lemma 8. We begin by explaining the high-level approach which will make one final composition of both F_ℓ and U_ℓ into subarrays. For any $j \geq 0$, let $U_\ell^j = U_\ell[\ell \cdot j, \ell \cdot (j+1) - 1]$ i.e. U_ℓ^j is the j-th consecutive ℓ-length subarray of U_ℓ. The key property that we will need is given in Lemma 9 which intuitively states that given half of the bits in U_ℓ, we can compute the other half with certainty.

▶ **Lemma 9.** *Let U_ℓ be chosen arbitrarily from $\{0101, 1010\}^{\frac{\ell}{4}}$. Given F_ℓ, $F_\ell \odot U_\ell$ and U_ℓ^{2j+1} for all $j \geq 0$, it is possible to uniquely determine U_ℓ^{2j} for all $j \geq 0$.*

We briefly justify why Lemma 8 is in-fact a straight-forward corollary of Lemma 9. If we pick U_ℓ uniformly at random from $\{0101, 1010\}^{\frac{\ell}{4}}$ then by Lemma 9, the conditional entropy, $H(F_\ell \odot U_\ell \,|\, U_\ell^{2j+1}$ for all $j)$ is $\Omega(\ell \lg \ell)$. This is because we always recover $\Theta(\lg \ell)$ distinct U_ℓ^{2j}, each of which is independent and has entropy $\Omega(\ell)$ bits. It then immediately follows that $H(F_\ell \odot U_\ell) \geq H(F_\ell \odot U_\ell \,|\, U_\ell^{2j+1}$ for all $j)$ as required. We also require for Lemma 8 that F_ℓ

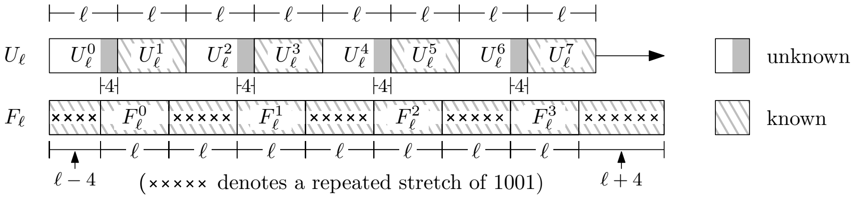

Figure 1 We can determine $U_\ell^{2j}[\ell-4,\ell-1]$ if we know F_ℓ, every U_ℓ^{2j+1} and $F_\ell \odot U_\ell$.

contains an equal number ones and zeros. This follows immediately from the description of F_ℓ below.

4.2 The subarray F_ℓ

We now give the description of F_ℓ which requires one final decomposition into subarrays which is also shown in Figure 1 below. For each $j \in [\lfloor(\lg \ell)/2\rfloor]$, F_ℓ contains a subarray F_ℓ^j of length ℓ. Intuitively, each subarray F_ℓ^j will be responsible for recovering U_ℓ^{2j}. These subarrays occur in order in F_ℓ. Before and after each F_ℓ^j there are stretches of repeats of the string 1001. Specifically, before F_ℓ^1 there are $\ell/4-1$ repeats the string 1001. Between each F_ℓ^j and F_ℓ^{j+1} there are $\ell/4$ repeats of the string 1001 and after $F_\ell^{\lfloor(\lg \ell)/2\rfloor-1}$ there are $\ell/4+1$ repeats. These repeats of 1001 are simply for structural padding and as we will see the contribution of these repeated 1001 strings to the L_2-rearrangement distance is independent of U_ℓ. This follows because the cost of permuting 1001 into 1010 or 0101 is always 2.

Finally, the structure of F_ℓ^j is as follows $F_\ell^j = 10^{(2^j+3)}1^{(\ell/4-1)}0^{(\ell/4-(2^j+3))}$. Here 0^z (resp. 1^z) is a string of z zeros (resp. ones). Intuitively, the reason that the stretch of 0s at the start of F_ℓ^j is the exponentially increasing with j is so that the number of positions the second one in F_ℓ^j (immediately after the stretch of 0s) is forced to move is also exponentially increasing with j as demonstrated in Figure 2 below. This is will allow us to recover each U_ℓ^{2j} from a different bit in the outputs. This will claim will be made precise in the proof below.

4.3 Recovering half of the updates – the proof of Lemma 9

We are now in a position to prove Lemma 9. Our main focus will be on first proving that given F_ℓ, U_ℓ^{2j+1} for all j and $F_\ell \odot U_\ell$, we can uniquely determine $U_\ell^{2j}[\ell-4,\ell-1]$ for each $j \geq 0$. That is, for each j whether the last four symbols of U_ℓ^{2j} are 0101 or 1010. This is shown diagrammatically in Figure 1. We will then argue that by a straight-forward repeated application of this argument we can in-fact recover the whole of U_ℓ^{2j} for all $j \geq 0$.

We will begin by making some simplifying observations about $(F_\ell \odot U_\ell)[0]$. Recall that $(F_\ell \odot U_\ell)[0]$ was defined to be the L_2-rearrangement distance between $F_\ell[0,|U_\ell|-1]$ and U_ℓ. The first observation is that the distance is finite because both strings contain an equal number of zeros and ones.

The L_2-rearrangement distance $(F_\ell \odot U_\ell)[0]$ can be expressed as the sum of the *contributions* from moving each $U_\ell[i]$, over all $i \in [m]$. Let the contribution of $U_\ell[i]$, denoted, $\mathrm{CT}(i)$ be the square of the number of positions that $U_\ell[i]$ is moved by under the optimal permutation. We then have that $(F_\ell \odot U_\ell)[0] = \sum_i \mathrm{CT}(j)$. Finally, we let $D^\star$ be the sum of the contributions of the locations in every $U_\ell^{2j}[\ell-4,\ell-1]$, i.e. $D^\star = \sum_j \sum_{k=0}^{3} (\mathrm{CT}(2j\cdot\ell+(\ell-4)+k)$. We will also refer to the contribution of a substring which is defined naturally to be the sum of the contributions of its constituent characters. For example the contribution of the substring U_ℓ^j is equal to $\sum\{\mathrm{CT}(r) \,|\, r \in [\ell\cdot j, \ell\cdot(j+1)-1]\}$.

Our proof will be in two stages. First we will prove in Lemma 10 that we can compute $D^\star$ from F_ℓ, $F_\ell \odot U_\ell$ and U_ℓ^{2j+1} for all $j \geq 0$. Second we will prove that for any $j > 0$, we can determine $U_\ell^{2j}[\ell-4, \ell-1]$ from $D^\star$.

In the proof of Lemma 10 we argue that $D^\star$ can be calculated directly from $(F_\ell \odot U_\ell)[0]$ by subtracting the contributions of U_ℓ^{2j+1} and $U_\ell^{2j}[0, \ell-5]$ for all $j \geq 0$. More specifically, we will prove that the contribution of any U_ℓ^{2j+1} can calculated from U_ℓ^{2j+1} and F_ℓ, which are both known. In particular, the contribution of any U_ℓ^{2j+1} is independent of every unknown U_ℓ^{2j}. Further, we will prove that although U_ℓ^{2j} is unknown, the contribution of $U_\ell^{2j}[0, \ell-5]$, always equals $\ell/2 - 2$, regardless of the choice of U_ℓ.

▶ **Lemma 10.** *$D^\star$ can be computed from F_ℓ, $F_\ell \odot U_\ell$ and U_ℓ^{2j+1} for all $j \geq 0$.*

Proof. In this proof we rely heavily on Lemma 3.1 from [1] which states that under the optimal permutation, the i-th one (resp. zero) in U_ℓ is moved to the i-th one (resp. zero) in $F_\ell[0, |U_\ell|-1]$. For any j, consider, U_ℓ^{2j} and U_ℓ^{2j+1}. The number of ones in U_ℓ^{2j} (resp. U_ℓ^{2j+1}) is fixed, independent of the choice of U_ℓ. In particular there are exactly $\ell/2$ zeros and $\ell/2$ ones. It is easily verified that, by construction, $F_\ell[2j \cdot \ell, (2j+2) \cdot \ell - 1]$ also contains exactly ℓ zeros and ℓ ones. Therefore, the i-th one (resp. zero) in U_ℓ^{2j} is moved to the i-th one (resp. zero) in $F_\ell[2j \cdot \ell, (2j+2) \cdot \ell - 1]$. Similarly, the i-th one (resp. zero) in U_ℓ^{2j} is moved to the $(i + \ell/2)$-th one (resp. zero) in $F_\ell[2j \cdot \ell, (2j+2) \cdot \ell - 1]$

Consider any U_ℓ^{2j+1} which is known. By the above observation, we can therefore determine which position in $F_\ell[2j \cdot \ell, (2j+2) \cdot \ell - 1]$, each character in U_ℓ^{2j+1} is moved to under the optimal permutation. From this we can then directly compute the contribution of each U_ℓ^{2j+1} to $(F_\ell \odot U_\ell)[0]$.

Consider any U_ℓ^{2j} which is unknown. As observed above, the i-th one (resp. zero) in U_ℓ^{2j} is moved to the i-th one (resp. zero) in $F_\ell[2j \cdot \ell, (2j+2) \cdot \ell - 1]$. By construction, we have that $F_\ell[2j \cdot \ell, (2j+1) \cdot \ell - 5]$ consists entirely of repeats of 1001. Further for any i, we have that $U_\ell^{2j}[4i, 4i+3]$ is either 1010 or 0101. Therefore for all $i < \ell/4$ we have that the two ones (resp. zeros) in $U_\ell^{2j}[4i, 4i+3]$ are moved to the two ones (resp. zeros) in $F_\ell[2j \cdot \ell + 4i, 2j \cdot \ell + 4i + 3] = 1001$. The key observation is that regardless of whether $U_\ell^{2j}[4i, 4i+3] = 1010$ or 0101, the contribution of $U_\ell^{2j}[4i, 4i+3]$ is 2. Therefore for any U_ℓ, the contribution of $U_\ell^{2j}[0, \ell-5]$ is always $\ell/2 - 2$.

Finally, the value of $D^\star$ is can be calculated directly from $(F_\ell \odot U_\ell)[0]$ as claimed by subtracting the calculated contributions of U_ℓ^{2j+1} and $U_\ell^{2j}[0, \ell-5]$ for all $j \geq 0$. ◀

In Lemma 12 we will prove that we can compute $U_\ell^{2j}[\ell-4, \ell-1]$ from $D^\star$ (for any sufficiently large j). The intuition behind this is given by Lemma 11 which gives an explicit formula for the contribution of $U_\ell^{2j}[\ell-4, \ell-1]$. Observe that the contribution depends only on whether $U_\ell^{2j}[\ell-4, \ell-1]$ equals 1010 ($v_j = 1$) or 0101 ($v_j = 0$). In the proof we begin by arguing that under the optimal permutation, the two ones (resp. zeros) in $U_\ell^{2j}[\ell-4, \ell-1]$ are moved to the leftmost two ones (resp. zeros) in F_ℓ^j as illustrated in Figure 2. The key observation is that regardless of whether v_j equals 0 or 1, by construction the right one in $U_\ell^{2j}[\ell-4, \ell-1]$ is moved exponentially far (as a function of j). Furthermore, in the $v_j = 1$ case the right one moves one position further than in the $v_j = 0$ case. As the contribution is the square of the number of positions a character moves, this creates a exponentially large change in the contribution. The exact contribution given in the Lemma can be calculated straightforwardly by considering each of the four symbols in $U_\ell^{2j}[\ell-4, \ell-1]$ individually.

▶ **Lemma 11.** *For any j, let $v_j = 1$ if $U_\ell^{2j}[\ell-4, \ell-1] = 1010$ and $v_j = 0$ otherwise. The contribution of $U_\ell^{2j}[\ell-4, \ell-1]$ is exactly $v_j \cdot 2^{j+1} + 2^{2j} + 2$.*

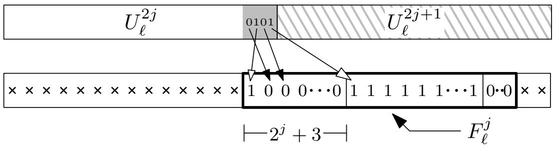

Figure 2 The permutation of the symbols in $U_\ell^{2j}[\ell-4,\ell-1]$ under the optimal permutation. The highlighted region is F_ℓ^j.

We can now prove Lemma 12 which follows almost immediately from Lemma 11.

▶ **Lemma 12.** *For any $j \geq 0$, it is possible to compute $U_\ell^{2j}[\ell-4,\ell-1]$ from $D^\star$.*

Proof. Let $D_2^\star$ equal $D^\star - \sum_j (2^{2j}+2)$ which can be calculated directly from $D^\star$. An alternative and equivalent definition of $D_2^\star$ follows from Lemma 11 and is given by $D_2^\star = \sum_j v_j \cdot 2^{j+1}$. We can therefore compute v_j and hence $U_\ell^{2j}[\ell-4,\ell-1]$ by inspecting the $(j+1)$-th bit in the binary representation of $D_2^\star$. ◀

Recall from Lemma 10 that $D^\star$ can in turn be computed from F_ℓ, $F_\ell \odot U_\ell$ and U_ℓ^{2j+1} for all $j \geq 0$. Therefore as claimed, given F_ℓ, $F_\ell \odot U_\ell$ and U_ℓ^{2j+1} for all $j \geq 0$, we can uniquely determine $U_\ell^{2j}[\ell-4,\ell-1]$ for each $j \geq 0$. Lemma 9 now follows almost immediately by repeat application of this argument as we now set out.

Recovering the rest of $U_{\ell,(2j)}$

So far we have only proven that we can recover $U_\ell^{2j}[\ell-4,\ell-1]$ for all j. The claim that we can in-fact recover the whole of U_ℓ^{2j} follows by repeatedly application of the argument above. Specifically, once we have recovered $U_\ell^{2j}[\ell-4,\ell-1]$ for all j, we can use this additional information (and $(F_\ell \odot U_\ell)[1]$ instead of $(F_\ell \odot U_\ell)[0]$) to recover $U_\ell^{2j}[\ell-8,\ell-5]$ for all j and so on. More formally we proceed by induction on increasing k by observing that using the above argument given F_ℓ, $(F_\ell \odot U_\ell)[k]$, U_ℓ^{2j+1} for all $j \geq 0$ and $U_\ell^{2j+1}[\ell-4k,\ell-1]$ for all $j \geq 0$ we can recover $U_\ell^{2j+1}[\ell-4k-4,\ell-4k-1]$ for all j.

References

1 A. Amir, Y. Aumann, G. Benson, A. Levy, O. Lipsky, E. Porat, S. Skiena, and U. Vishne. Pattern matching with address errors: rearrangement distances. In *SODA'06: Proc. 17th ACM-SIAM Symp. on Discrete Algorithms*, pages 1221–1229. ACM Press, 2006.

2 A. Amir, Y. Aumann, G. Benson, A. Levy, O. Lipsky, E. Porat, S. Skiena, and U. Vishne. Pattern matching with address errors: Rearrangement distances. *Journal of Computer System Sciences*, 75(6):359–370, 2009.

3 R. Clifford and M. Jalsenius. Lower bounds for online integer multiplication and convolution in the cell-probe model. In *ICALP'11: Proc. 38th International Colloquium on Automata, Languages and Programming*, pages 593–604, 2011. arXiv:1101.0768.

4 R. Clifford, M. Jalsenius, and B. Sach. Tight cell-probe bounds for online hamming distance computation. In *SODA'13: Proc. 24th ACM-SIAM Symp. on Discrete Algorithms*, pages 664–674, 2013. arXiv:1207.1885.

5 R. Clifford, M. Jalsenius, and B. Sach. Cell-probe bounds for online edit distance and other pattern matching problems. In *SODA'15: Proc. 26th ACM-SIAM Symp. on Discrete Algorithms*, 2015. arXiv:1407.6559.

6 R. Clifford and B. Sach. Pattern matching in pseudo real-time. *Journal of Discrete Algorithms*, 9(1):67–81, 2011.

7 M. Fredman. Observations on the complexity of generating quasi-Gray codes. *SIAM Journal on Computing*, 7(2):134–146, 1978.

8 M. Fredman and M. Saks. The cell probe complexity of dynamic data structures. In *STOC'89: Proc. 21st Annual ACM Symp. Theory of Computing*, pages 345–354, 1989.

9 K. Green Larsen. The cell probe complexity of dynamic range counting. In *STOC'12: Proc. 44th Annual ACM Symp. Theory of Computing*, pages 85–94, 2012.

10 K. Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *FOCS'12: Proc. 53rd Annual Symp. Foundations of Computer Science*, pages 293–301, 2012.

11 M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

12 M. Pătraşcu. *Lower bound techniques for data structures.* PhD thesis, MIT, 2008.

13 M. Pătraşcu and E. D. Demaine. Tight bounds for the partial-sums problem. In *SODA'04: Proc. 15th ACM-SIAM Symp. on Discrete Algorithms*, pages 20–29, 2004.

14 M. Pătraşcu and E. D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM Journal on Computing*, 35(4):932–963, 2006.

15 R.Clifford, A. Grønlund, and K. Green Larsen. New unconditional hardness results for dynamic and online problems. In *FOCS'15: Proc. 56th Annual Symp. Foundations of Computer Science*, pages 1089–1107, 2015.

16 A. Yao. Probabilistic computations: Toward a unified measure of complexity. In *FOCS'77: Proc. 18th Annual Symp. Foundations of Computer Science*, pages 222–227, 1977.

17 A. Yao. Should tables be sorted? *Journal of the ACM*, 28(3):615–628, 1981.