Theoretical Computer Science 483 (2013) 68–74

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Space lower bounds for online pattern matching

Raphaël Clifford [a], Markus Jalsenius [a], Ely Porat [b], Benjamin Sach [c,*]

[a] University of Bristol, Department of Computer Science, Bristol, UK
[b] Bar-Ilan University, Department of Computer Science, Ramat-Gan, Israel
[c] University of Warwick, Department of Computer Science, Coventry, UK

ABSTRACT

We present space lower bounds for online pattern matching under a number of different distance measures. Given a pattern of length m and a text that arrives one character at a time, the online pattern matching problem is to report the distance between the pattern and a sliding window of the text as soon as the new character arrives. We require that the correct answer is given at each position with constant probability. We give $\Omega(m)$ bit space lower bounds for L_1, L_2, L_∞, Hamming, edit and swap distances as well as for any algorithm that computes the cross-correlation/convolution. We then show a dichotomy between distance functions that have wildcard-like properties and those that do not. In the former case which includes, as an example, pattern matching with character classes, we give $\Omega(m)$ bit space lower bounds. For other distance functions, we show that there exist space bounds of $\Omega(\log m)$ and $O(\log^2 m)$ bits. Finally we discuss space lower bounds for non-binary inputs and show how in some cases they can be improved.

1. Introduction

We combine existing results with new observations to present an overview of space lower bounds for online pattern matching. Given a pattern that is provided in advance and a text that arrives one character at a time, the online pattern matching problem is to report the distance between the pattern and a sliding window of the text as soon as the new character arrives. In this formulation, the pattern is processed before the first text character arrives and, once processed, the pattern is no longer available to the algorithm unless a copy is explicitly made.

This problem has recently gained a great deal of interest with breakthrough results given for exact matching and pattern matching under bounded Hamming distance (k-mismatch) [13]. For both problems it was shown that space sublinear in the size of the pattern is sufficient to give the correct answer at every alignment with high probability. These remarkable results immediately raise a number of significant unresolved questions. The first is for which other distance measures between strings might sublinear space randomised online algorithms be achievable and it is this question which we address here.

Our presentation is divided between what we term local and non-local online pattern matching problems. In the former case the distance function between a pattern P of length m and an m-length substring of the text T, starting at position i, is defined by

$$\textsc{LocalPM}_{(\oplus,\Delta)}(P,T)=\bigoplus_{j=0}^{m-1}\Delta(P[j],T[i+j]),$$

[*] Corresponding author. Tel.: +44 7584667532.
E-mail address: ben@cs.bris.ac.uk (B. Sach).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.06.012

where $\oplus$ and Δ are both binary operators. In Section 4 we show $\Omega(m)$ bit space lower bounds for online pattern matching for the local problems of L_1, L_2, and Hamming distance as well as for any algorithm that computes the cross-correlation/convolution.

We then go on to show in Section 5 a space dichotomy for local online pattern matching problems of the form $d(i) = \bigwedge_{j=0}^{m-1} \Delta(P[j], T[i+j])$, where the range of Δ is {True, False}. Where the distance function Δ has wildcard-like properties (qv. Section 5), we give an $\Omega(m)$ space lower bound. Where it does not, we have $\Omega(\log m)$ and $O(\log^2 m)$ space bounds. This implies, for example, that online pattern matching with character classes [8] requires linear space.

In Section 6 we go on to consider all eight possible binary Boolean associative operators and give a complete classification in terms of their known upper and lower space bounds. One consequence is that determining if there is an exact "non-match", where the Hamming distance is the same as the pattern length, requires linear space in our online model. This bound also holds if, for example, only the parity of the Hamming distance is required. In Section 7 we then show how our techniques can be used to give linear space lower bounds for L_∞ online pattern matching. In Section 8 we discuss a possible approach to space lower bounds for inputs with large alphabets, focussing on the Hamming distance problem. Finally, in Section 9 we explore non-local problems and show $\Omega(m)$ bit space lower bounds for both online edit and swap distance.

2. Preliminaries and related work

Let Σ_{P} and Σ_{T} denote the pattern and text alphabet, respectively. We say that LocalPM$_{(\oplus,\Delta)}$ is *text independent* with respect to the pattern P if the value of LocalPM$_{(\oplus,\Delta)}$ is a constant independent of T. We say that LocalPM$_{(\oplus,\Delta)}$ is *pattern independent* with respect to a pattern P if there is a function Δ' such that $\Delta(x, y) = \Delta'(y)$ for all $(x, y) \in \Sigma_{\mathrm{P}} \times \Sigma_{\mathrm{T}}$.

Example 1. Let $\Sigma_{\mathrm{P}} = \{x, y, z\}$, $\Sigma_{\mathrm{T}} = \{a, b, c\}$, $\oplus$ be the Boolean AND-operator and Δ be defined according to the table in Fig. 1, where 1 is True and 0 is False. We can see that LocalPM$_{(\wedge,\Delta)}$ is text independent with respect to the pattern $P = xxyyxzxx$ as it always outputs 0. It is also pattern independent with respect to $P = yyzyyzzy$ as $\Delta(y, \alpha) = \Delta(z, \alpha)$ for all $\alpha \in \Sigma_{\mathrm{T}}$. In fact, for this particular definition of Δ, LocalPM$_{(\wedge,\Delta)}$ is either text or pattern independent with respect to any pattern P.

Suppose that LocalPM$_{(\oplus,\Delta)}$ is text independent with respect to a pattern P. Then any algorithm for LocalPM$_{(\oplus,\Delta)}$ on P requires at most $O(1)$ space after preprocessing P. If LocalPM$_{(\oplus,\Delta)}$ is pattern independent with respect to P then LocalPM$_{(\oplus,\Delta)}$ does not depend on the pattern and is outside the scope of this paper.

We say that LocalPM$_{(\oplus,\Delta)}$ is *invalid* if, for every pattern P, it is either text or pattern independent with respect to P. LocalPM$_{(\oplus,\Delta)}$ is *valid* if it is not invalid. The problem LocalPM$_{(\wedge,\Delta)}$ in the previous example is therefore invalid. We will only consider from this point pattern matching problems LocalPM$_{(\oplus,\Delta)}$ which are valid, and ignore patterns for which LocalPM$_{(\oplus,\Delta)}$ is pattern or text independent.

Our focus is on online pattern matching algorithms which output correct answers with constant probability. We are not aware of previous work that considers randomised lower bounds for this specific type of problem. There is however now a considerable literature on communication complexity and on streaming algorithms for single input streams, including those that process a sliding window of the input (see e.g. [4]). This previous streaming work has typically focussed on deterministic or randomised bounds for finding approximate rather than exact solutions. Quantum lower and classical upper bounds for the communication complexity of Hamming distance in more general models than we consider were given previously [5]. A linear lower bound for the randomised communication complexity of the inner product of two binary vectors is given in [3]. The dichotomy presented in Section 5 and in particular the concept of a matching relation that includes wildcard matching, although in a different setting and with different terminology, is similar to a time complexity dichotomy given previously by Muthukrishnan and Ramesh [9]. On the topic of swap matching in Section 9, we note that in [1], the existence of a reduction for time rather than space, from Boolean convolutions to string matching with swaps is claimed without proof.

3. Communication complexity problems

Our results are based on reductions from various one-way randomised communication complexity problems with known lower bounds. We list the relevant problems below. In a one-way randomised communication model, only Alice can send messages to Bob and Bob must output the correct answer with probability at least 2/3. Note that the value 2/3 is inconsequential: any probability strictly greater than 1/2 can be amplified to a constant arbitrary close to 1. We assume private randomness.

Definition 2. The Equality problem in one-way communication complexity is defined as follows. Alice has a string $X \in \{0, 1\}^m$ and Bob has a string $Y \in \{0, 1\}^m$. Bob must determine whether $X = Y$. The randomised communication complexity is $\Theta(\log m)$ bits [14].

Definition 3. The Indexing problem in one-way communication complexity is defined as follows. Alice has a string $X \in \{0, 1\}^m$ and Bob has an index $n \in \{0, \ldots m-1\}$. Bob must find $X[n]$. The problem is known to have an $\Omega(m)$ bit lower bound (see [6] for an elementary proof).

Δ	a	b	c
x	0	0	0
y	0	1	1
z	0	1	1

Fig. 1. An example of Δ such that LocalPM$_{(\wedge,\Delta)}$ is invalid (either text or pattern independent with respect to any pattern P).

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Fig. 2. The wildcard matrix (left) and negated wildcard matrix (right).

Δ	a	b
$\star$	1	1 .
x	1	0

Fig. 3. Δ in the proof of Theorem 6.

4. Addition

In this section we consider the problem LocalPM$_{(+,\Delta)}$, where $+$ is standard addition and the range of Δ is a subset of the integers. That is, the distance function is

$$d(i) = \sum_{j=0}^{m-1} \Delta(P[j], T[i+j]).$$

Theorem 4. LocalPM$_{(+,\Delta)}$ *requires $\Omega(m)$ bits of space.*

Proof. Since LocalPM$_{(+,\Delta)}$ is not text independent, there must exist characters $x \in \Sigma_{\mathrm{P}}$ and $a, b \in \Sigma_{\mathrm{T}}$ such that $\Delta(x, a) \neq \Delta(x, b)$. We reduce from Indexing: Alice has a string $T = \{a, b\}^m$ and Bob has an index n. Alice initialises a pattern matching algorithm A on the pattern $P = \{x\}^m$ and feeds in her string T. Then she sends the internal state of A to Bob, who feeds in n copies of the symbol a. Let d be the output after those as. Bob then feeds in another a. Let d' be the output. If $d = d'$ then $A[n] = a$. If $d \neq d'$ then $A[n] = b$. If the probability of error per output is bounded by a constant $c < 1/4$, then the union bound for error on two outputs is $2c$, giving the Indexing problem an error probability of at most $2c < 1/2$. □

Corollary 5. *Computing the L_1, L_2 and Hamming distances, as well as the convolution, require $\Omega(m)$ bits of space.*

5. Conjunction

In this section we consider LocalPM$_{(\wedge,\Delta)}$, where $\wedge$ is the Boolean AND-operator and the range of Δ is $\{0, 1\}$ (where 0 denotes False and 1 denotes True). There are several natural pattern matching problems that fall under this category, for example, exact matching, matching with wildcards and exact matching with character classes.

The function Δ can be represented with a 0/1-matrix M_Δ, where the rows and columns correspond to the symbols in Σ_{P} and Σ_{T}, respectively. Thus, the entry $(i, j) = \Delta(i, j)$. The 2×2 matrix in Fig. 2 will play an important role, and we call it the *wildcard matrix*.

We say that M_Δ contains the wildcard matrix if it is a submatrix of M_Δ under some permutation of the rows and columns.

We demonstrate the following dichotomy for LocalPM$_{(\wedge,\Delta)}$. If M_Δ contains the wildcard matrix, then LocalPM$_{(\wedge,\Delta)}$ is solvable in $\tilde{\Theta}(m)$ bits of space, otherwise it is solvable in $\tilde{\Theta}(1)$ bits of space. The first class is equivalent to pattern matching with wildcards, and the second class is equivalent to exact matching. Note that both dichotomies are decidable due to the simple characteristic of the function Δ.

Theorem 6. *If M_Δ contains the wildcard matrix, then* LocalPM$_{(\wedge,\Delta)}$ *requires $\Omega(m)$ bits of space.*

Proof. Suppose that $\star, x \in \Sigma_{\mathrm{P}}$ ($\star$ represents a wildcard symbol) and $a, b \in \Sigma_{\mathrm{T}}$ such that Δ is specified according to Fig. 3. We reduce from the Indexing problem, in which Alice has an m-length bit string $X \in \{\star, x\}^m$ and Bob has an index $n \in \{0, \dots m-1\}$. Let the pattern P be the string X. Let A be any algorithm that solves LocalPM$_{(\wedge,\Delta)}$ on the pattern P. Alice sends the internal state of A to Bob, who feeds the algorithm with the m-length string that has the symbol a at every position except for at position n where the symbol is b. The output is True if and only if $X[n] = \star$. □

The following lemma will be useful for the next two theorems (see Fig. 4).

M_Δ	a	b	c	d	e	f
v	0	1	0	1	1	0
w	0	0	0	0	0	0
x	1	0	1	0	0	0
y	0	1	0	1	1	0
z	1	0	1	0	0	0

M'_Δ	a	b	c	d	e	f
v	0	1	–	–	–	–
w	–	–	–	–	–	–
x	1	0	–	–	–	–
y	–	–	–	–	–	–
z	–	–	–	–	–	–

Id.	a	b
x	1	0
v	0	1

Fig. 4. An illustration of Lemma 7.

Δ	a	b
x	1	0
y	0	1

Fig. 5. Δ in the proof of Theorem 8.

Lemma 7. *Let M'_Δ be the matrix obtained from M_Δ by first removing copies of identical rows and columns, keeping only rows and columns that are distinct in M_Δ, and then removing any row or column that contains only zeros. If M_Δ does not contain the wildcard matrix, then M'_Δ is the identity matrix, under some permutation of rows and columns.*

Proof. Suppose that M_Δ does not contain the wildcard matrix. Let M'_Δ be obtained from M_Δ according to the statement of the lemma. We will show that every column and every row of M'_Δ contains exactly one 1.

First we show that every row of M'_Δ must contain at least one 1. Suppose that some row r of M'_Δ contains only 0s. Since zero-rows of M_Δ were removed and one copy of each column remains after the removal process, it is not possible that all columns in which row r is 1 were removed. We now show that M'_Δ cannot contain a row r with two or more 1s. Without loss of generality, assume that there is a 1 in columns i and j of row r. Since M_Δ does not contain a wildcard matrix, the elements of columns i and j must both be either 0 or 1 in every row. Thus, columns i and j are identical, and one of them must have been removed, contradicting the fact that there are two 1s in row r of M'_Δ. In order to show that every column of M'_Δ contains exactly one 1, we use the exact same argument as for the rows. Thus, M'_Δ is the identity matrix, under some permutation of rows and columns. (See Fig. 4 for an illustration of the lemma.) □

Theorem 8. *If M_Δ does not contain the wildcard matrix, then* LocalPM$_{(\wedge,\Delta)}$ *requires $\Omega(\log m)$ bits of space.*

Proof. We reduce from the Equality problem, where Alice has a string $X \in \{0,1\}^m$ and Bob has a bit string $Y \in \{0,1\}^m$. Since M_Δ doesn't contain the wildcard matrix and as we only consider problems LocalPM$_{(\wedge,\Delta)}$ that are valid, it follows from Lemma 7 that there must exist $x, y \in \Sigma_\text{P}$ and $a, b \in \Sigma_\text{T}$ such that Δ is according to Fig. 5. Let P be the m-length pattern obtained from X by replacing every 0 with x and every 1 with y. The m-length text T is obtained similarly from Y by replacing every 0 with a and every 1 with b. For any algorithm A that solves LocalPM$_{(\wedge,\Delta)}$ on the pattern P, Alice sends the internal state of A on pattern P to Bob, who feeds A with T. The output is True if and only if $X = Y$. □

Theorem 9. *If M_Δ does not contain the wildcard matrix, then* LocalPM$_{(\wedge,\Delta)}$ *can be solved in $O(\log^2 m)$ bits of space.*

Proof. We will describe an algorithm for solving LocalPM$_{(\wedge,\Delta)}$ which uses the exact matching algorithm by Porat and Porat [13], which runs in space $O(\log m)$ words, which is $O(\log^2 m)$ bits of space (under the word-RAM model). In order to use the exact matching algorithm (as a "black box") we must ensure that we do not feed it with distinct symbols that are identical under Δ. In other words, we can think of Δ specifying character classes, and for each class we want to use one representative symbol. We formalise this below.

We make the very reasonable assumption that the alphabets Σ_P and Σ_T are both enumerable and that we can iterate through every symbol of Σ_P and Σ_T, respectively, in no more than $O(\log m)$ bits of space. Let the order by which we iterate through the alphabets describe an ordering of the symbols in Σ_P and Σ_T. We say that the symbol $x \in \Sigma_\text{P}$ is *smaller* than $y \in \Sigma_\text{P}$ if x appears before y when iterating through Σ_P. We use the same notation for the symbols of Σ_T. We say that two symbols $x, y \in \Sigma_\text{P}$ are *equivalent* if $\Delta(x, a) = \Delta(y, a)$ for all $a \in \Sigma_\text{T}$. Similarly, $a, b \in \Sigma_\text{T}$ are equivalent if $\Delta(x, a) = \Delta(x, b)$ for all $x \in \Sigma_\text{P}$. We define the *smallest equivalent* symbol of $x \in \Sigma_\text{P}$ to be the symbol $y \in \Sigma_\text{P}$ such that y is equivalent to x and no other symbol equivalent to x is smaller than y. The notion of smallest equivalent symbol is defined similarly on Σ_T.

Let $\Sigma'_\text{P} \subseteq \Sigma_\text{P}$ be the set of all symbols $x \in \Sigma_\text{P}$ such that the smallest equivalent symbol of x is x itself. We do not include any symbol x in Σ'_P such that $\Delta(x, a) = 0$ for all $a \in \Sigma_\text{T}$. Similarly, let $\Sigma'_\text{T} \subseteq \Sigma_\text{T}$ be the set of all symbols $a \in \Sigma_\text{T}$ such that the smallest equivalent symbol of a is a itself. We do not include any symbol a in Σ'_T such that $\Delta(x, a) = 0$ for all $x \in \Sigma_\text{P}$. By Lemma 7 we have that Δ on Σ'_P and Σ'_T is represented by an identity matrix under some permutation of the rows and columns. In the example of Fig. 4, $\Sigma'_\text{P} = \{x, v\}$ and $\Sigma'_\text{T} = \{a, b\}$. We will ensure that we use the exact matching algorithm of [13] only on Σ'_P and Σ'_T (i.e., normal exact pattern matching).

Given a symbol $x \in \Sigma_\text{P}$, we can find its smallest equivalent symbol by iterating through every symbol $y \in \Sigma_\text{P}$ and for each y, we iterate through all $a \in \Sigma_\text{T}$ to check whether $\Delta(x, a) = \Delta(y, a)$. Similarly we can find the smallest equivalent symbol of any symbol in Σ_T.

Let P be the pattern. We may assume that P does not contain a symbol x for which $\Delta(x, a) = 0$ for all $a \in \Sigma_T$. If it does, the output is always 0. Before we preprocess the pattern, we replace every symbol with its smallest equivalent symbol. Then we preprocess the pattern using the fingerprint technique described in [13]. Now we run the exact matching algorithm with the following additional step. When a new symbol a arrives, we replace it with its smallest equivalent symbol. The only caveat we must take care of is the situation when $\Delta(x, a) = 0$ for all $x \in \Sigma_P$. We can detect this case by iterating through the symbols of Σ_P. As long as a is present in the last m characters of the stream, the output is zero. We use a flag to keep track of this. □

We now show how these results can be applied to a specific pattern matching problem that has not been considered in the online setting before. The pattern matching with character classes problem allows a set of characters to be defined for each position in the pattern [8]. A character in the text matches a set at a pattern position if it is contained within it. This is a generalisation of exact matching where each set would contain only one character. Using Theorems 6, 8 and 9 we can determine precisely when this problem can and cannot be solved online in sublinear space.

Corollary 10. *Online pattern matching with character classes requires $\Omega(m)$ bits of space in the worst case. However, where the character classes define a matching relation Δ which does not contain the wildcard matrix (see the example in Fig. 4), $O(\log^2 m)$ bits suffice.*

6. Other Boolean operators

In the previous section we demonstrated a dichotomy for $\text{LocalPM}_{(\oplus,\Delta)}$, where $\oplus$ is the AND-operator. Here we will complete the classification of Boolean operators. There are eight associative Boolean operators $a \oplus b$:

1. True **2.** False **3.** a **4.** b **5.** $a \wedge b$ **6.** $a \vee b$ **7.** $a = b$ **8.** $a \neq b$.

The operators True and False are trivial; the output is either always True or False. The operator $a \oplus b = b$ is also easy; the output is always $\Delta(P[m-1], t)$, where t is the last received symbol of the text stream.

The operator $a \oplus b = a$ is on the other hand more demanding. Here the output is $\Delta(P[0], t)$, where t is the mth last symbol received from the text stream. The pattern matching algorithm must therefore remember m received characters of the stream. More precisely, we see that $\Omega(m)$ bits of space is necessary by reducing from the Indexing problem: Alice first feeds her array (text) into the pattern matching algorithm, for which $P[0]$ is a character that can distinguish between the characters of Alice's array. She then sends the internal state to Bob, who feeds in n symbols in order to determine the value at index n of Alice's array.

The OR-operator $\vee$ is equivalent to $\wedge$ under De Morgan's laws: negate the outputs from Δ and negate the output from the pattern matching algorithm. Thus, the dichotomy for $\wedge$ applies to $\vee$ as well, only that we characterise the classes with the wildcard matrix in which each element has been negated. This is called the negated wildcard matrix (see Fig. 2).

We now show that the equality operator "=" requires $\Omega(m)$ bits of space. First note that the output from the pattern matching algorithm is 0 if and only if $\Delta([P[j], T[i+j]) = 0$ for an odd number of positions j. For example, if M_Δ is the identity matrix, $\text{LocalPM}_{(=,\Delta)}$ gives us the parity of the Hamming distance.

Since $\text{LocalPM}_{(=,\Delta)}$ is valid, there are $x \in \Sigma_P, a, b \in \Sigma_T$ such that $\Delta(x, a) = 0$ and $\Delta(x, b) = 1$. We reduce from the Indexing problem, where Alice has a string in $\{a, b\}^m$ and Bob has an index n. Alice initialises a pattern matching algorithm on the pattern $P = \{x\}^m$ and feeds it with her string. She sends the internal state to Bob, who feeds the algorithm with n copies of the symbol a. The first position of P is now aligned with the nth character of Alice's string. Suppose the output from the algorithm is d. Bob now feeds in another a. Let d' be the new output. If $d = d'$ then the character at position n of Alice's string must have been a. If $d \neq d'$ then the character must have been b.

The operator "$\neq$" is similar to "=" and also requires $\Omega(m)$ bits of space. To see this, note that the output from the pattern matching algorithm is 0 if and only if $\Delta([P[j], T[i+j]) = 1$ for an even number of positions j. We may therefore prove the lower bound using a reduction from the Indexing problem similar to above.

7. The L_∞ distance

In this section we consider the L_∞ distance problem which can be defined as $\text{LocalPM}_{(\max,\Delta)}$, where $\Delta(x, y) = |x - y|$ and $\max(a, b)$ is the maximum of a and b. In this section we assume that the pattern and text are integer valued. Here the distance function is the maximum $\Delta(P[j], T[i+j])$ over all j, that is

$$d(i) = \max_{j\in\{0,\dots,m-1\}} \Delta(P[j], T[i+j]) .$$

Theorem 11. *The L_∞ distance problem requires $\Omega(m)$ bits of space.*

Proof. Let $\Sigma_P = \{0, 1\}$ and $\Sigma_T = \{2, 3\}$. Therefore Δ is specified according to Fig. 6. Let $\Delta'(x, y) = 1$ if $\Delta(x, y) < 3$, otherwise $\Delta'(x, y) = 0$. Therefore $M_{\Delta'}$ contains the wildcard matrix and hence by Theorem 6, $\text{LocalPM}_{(\wedge,\Delta')}$ requires $\Omega(m)$ space.

Let $d'(i)$ be the distance under $\text{LocalPM}_{(\wedge,\Delta')}$. If $d'(i) = 1$ then for all j, $\Delta'(P[j], T[i+j]) = 1$, implying that $\Delta(P[j], T[i+j]) < 3$ for all j. Hence $d(i) < 3$. If $d'(i) = 0$ then there exists a j such that $\Delta'(P[j], T[i+j]) = 0$, implying that $\Delta(P[j], T[i+j]) = 3$ and hence $d(i) = 3$. Therefore, if we can solve $\text{LocalPM}_{(\max,\Delta)}$, we can solve $\text{LocalPM}_{(\wedge,\Delta')}$. □

Δ	2	3
1	1	2
0	2	3

Δ'	2	3
1	1	1
0	1	0

Fig. 6. Δ and Δ' in the proof of Theorem 11.

8. Non-binary alphabets

The space lower bounds we have given so far have been either $\Omega(\log m)$ or $\Omega(m)$ bits. When the pattern or text alphabet is drawn from a large universe, the question arises as to whether even more space is required to perform online pattern matching. We show by way of another different reduction a method that may be applicable to a wider range of pattern matching problems than we consider here. Our approach is to show a reduction from the communication complexity problem Disjointness [7] to the Hamming distance problem. In Disjointness Alice and Bob both have sets of m elements each chosen from a universe of size U and Bob wants to determine if their intersection is empty. The lower bound for the space complexity of the Hamming distance problem will then be determined by lower bounds for the one-way randomised communication complexity of the Disjointness problem with private coins. A result regarded as folklore shows that this complexity is $\Omega(m \log m + \log\log U)$ when U is $\Omega(m^{1+\varepsilon})$ [11,12]. This in turn implies a superlinear lower bound for the space complexity of the online Hamming distance problem with large alphabets.

For an integer n, we write $[n]$ to denote the set $\{0, \ldots, n-1\}$. Alice has a set $A \subseteq [U]$ and Bob has a set $B \subseteq [U]$, and $|A| = |B| = m$. The reduction performs the following steps. We assume for the moment that Alice and Bob both have a shared source of randomness and show later how this assumption can be removed.

1. Alice creates a pairwise independent hash function $h : [U] \to [cm]$, for some constant integer $c > 1$ and creates a pattern P of length cm where each element is initialised to be some unique symbol $\$ \notin [U]$. She then sets $P[h(x)] = x$ for all $x \in A$ by going through A in some arbitrary order. If a position of P is written to multiple times, only the last write is stored.
2. Alice starts the Hamming distance algorithm up until the point at which it has processed the pattern P but none of the text (which is created later) and sends the internal state of the algorithm to Bob.
3. Bob performs the same hashing operation using the same hash function but this time on set B, creating a text T of length cm. Bob uses a different unique symbol $\$' \notin [U]$ for the initialisation of the text.
4. Bob feeds the Hamming distance algorithm with the whole text T. Bob concludes that A and B are disjoint if and only if the output is cm.

Theorem 12. *Any randomised algorithm for Hamming distance where the symbols are chosen from a universe of size $\Omega(m^{1+\varepsilon})$ uses $\Omega(m \log m + \log\log U)$ bits of space.*

Proof. Considering the reduction above, if A and B are disjoint, then a deterministic Hamming distance algorithm will always output cm. If A and B are not disjoint then a necessary condition for a deterministic Hamming distance algorithm to output cm is if at least two elements are hashed to the same location by either Alice or Bob. We can see that the probability of incorrectly outputting cm is maximised when A and B share exactly one element. Therefore, suppose that $A \cap B = \{x\}$. The element x is hashed to position $h(x)$. By the union bound and the pairwise independence of the hash function, the probability that some other element in either A or B is mapped to $h(x)$ is at most $1/(cm) \cdot m \cdot 2 = 2/c$. If we assume our randomised Hamming distance algorithm is correct with probability at least $2/3$, then the overall process falsely reports disjointness with probability at most $2/c + 1/3$ (union bound). The space complexity of Hamming distance is therefore lower bounded by the communication complexity of the disjointness problem if Alice and Bob have a shared source of random bits to select their common hash function. By Newman's Theorem [10] the cost of transforming the protocol to work with only private coins is at most an additive $O(\log\log U)$ factor in the asymptotic complexity. Assuming that U grows polynomially in m and so $\log\log U$ is $O(\log m)$, the overall lower bound for the space complexity of the Hamming distance problem is therefore $\Omega(m \log m - \log m) = \Omega(m \log m)$. To finish the proof for larger U, we observe first that a lower bound for smaller universes must still hold for larger ones. The final additive $\Omega(\log\log U)$ term is derived by simply setting $m = 1$ and follows directly from the randomised lower bound for Equality. Therefore the overall lower bound is $\Omega(m \log m + \log\log U)$ as required. □

9. Non-local pattern matching

So far we have focused only on local pattern matching where each position in the alignment contributes to the distance independently of the other positions. Here we take a brief look at space lower bounds for two non-local distance measures: edit distance and swap matching.

In online pattern matching, we define the *edit distance* as the minimum number of single character edit operations (insert, delete and replace) required to transform P into the last m characters of the streamed text. This implies that the number of insertions and deletions are equal.

We show that for binary $\Sigma_{\mathrm{P}} = \Sigma_{\mathrm{T}} = \{0, 1\}$, the online edit distance problem requires $\Omega(m)$ bits of space. For non-binary inputs there is a reduction from the Hamming distance problem [2]. The reduction we give covers the binary alphabet case

Δ	a	b
★	1	1
x	1	0

a :	00**01**0	b :	0**10**00	b :	01000
★ :	00**10**0	★ :	0**01**00	x :	00010

Fig. 7. Δ and alignments under swaps.

as well and follows directly from Indexing, where Alice has a string $P \in \{0, 1\}^m$ and Bob has an index n. Alice initialises a pattern matching algorithm on the pattern P and sends the internal state to Bob, who first feeds in m zeros. Let d be the output and note that d is the number of ones in P. Bob then feeds in the m-length string that consists of zeros at every position except for at position n where it is one. Let d' be the output. Bob can now decide the value of $P[n]$ by comparing d with d': $P[n] = 1$ if $d' < d$, and $P[n] = 0$ if $d' \geqslant d$. The probability of error is therefore upper bounded by the union bound on d and d' being wrong.

Given a string S, a *swap* at position i means that the characters $S[i]$ and $S[i + 1]$ swap positions. We say there is a *swap match* if and only if the pattern P can be transformed into the last m characters of the streamed text through a set of swaps. Each $S[i]$ is swapped at most once.

We show that the online swap distance problem requires $\Omega(m)$ space. Our proof is based on the techniques we have presented in this paper. Specifically, we demonstrate a reduction from LocalPM$_{(\wedge,\Delta)}$ where M_Δ contains the wildcard matrix, hence the space lower bound is $\Omega(m)$. Suppose we have Δ as in Fig. 7. Let $P \in \{\star, x\}^m$ and $\Sigma_T = \{a, b\}$. From P we obtain $P' \in \{0, 1\}^{5m}$ such that every $\star$ in P is replaced with 00100 and every x is replaced with 00010. When we receive characters from the text, we replace a with 00010 and b with 01000. It follows, under the transformation of the symbols, that there is a swap match if and only if LocalPM$_{(\wedge,\Delta)}$ outputs True for the original (non-transformed) strings. To see this, note that both a and b, under the transformation, swap match $\star$, but b does not swap match x (see Fig. 7). The transformation of the symbols does not allow swaps between adjacent characters; every possible swap will take place "within" the binary encoding of a symbol. Thus, a swap match directly corresponds to a match under LocalPM$_{(\wedge,\Delta)}$.

10. Open problems

We have considered space lower bounds and discussed how they can be derived from known communication complexity lower bounds. Upper bounds can also be directly derived from existing online pattern matching algorithms. For all the problems we have discussed there is at most a log factor gap between these upper and lower bounds. However, where the known lower bound is sublinear, as is the case for exact matching for example, this gap may still be considered significant. Further, for bounded Hamming distance where the distance is only to be given if it is at most some constant k, the best known randomised online space upper bound is $O(k^3 \text{polylog}\, m)$ [13]). The best known lower bound, on the other hand, is very different at $\Omega(k)$ [5]. Further, it is known that the lower bounds cannot be increased to match the known upper bounds using the one-way communication complexity of the functions between two strings of the same length. Either more space efficient algorithms exist for these problems or novel techniques will be needed to improve the lower bounds.

References

[1] A. Amir, Y. Aumann, G. Landau, M. Lewenstein, N. Lewenstein, Pattern matching with swaps, Journal of Algorithms 37 (2000) 247–266.

[2] Ziv Bar-Yossef, T.S. Jayram, Robert Krauthgamer, Ravi Kumar, Approximating edit distance efficiently, in: FOCS'04: Proc. 45th Annual Symp. Foundations of Computer Science, 2004, pp. 550–559.

[3] B. Chor, O. Goldreich, Unbiased bits from sources of weak randomness and probabilistic communication complexity, SIAM Journal on Computing 17 (2) (1988) 230–261.

[4] M. Datar, A. Gionis, P. Inkyk, R. Motwani, Maintaining stream statistics over sliding windows, SIAM Journal on Computing 31 (6) (2002) 1794–1813.

[5] Wei Huang, Yaoyun Shi, Shengyu Zhang, Yufan Zhu, The communication complexity of the Hamming distance problem, Information Processing Letters 99 (4) (2006) 149–153.

[6] T.S. Jayram, Ravi Kumar, D. Sivakumar, The one-way communication complexity of Hamming distance, Theory of Computing 4 (1) (2008) 129–135.

[7] Eyal Kushilevitz, Noam Nisan, Communication Complexity, Cambridge University Press, 1997.

[8] Chaim Linhart, Ron Shamir, Faster pattern matching with character classes using prime number encoding, Journal of Computer System Sciences 75 (3) (2009) 155–162.

[9] S. Muthukrishnan, H. Ramesh, String matching under a general matching relation, Information and Computation 122 (1) (1995) 140–148.

[10] I. Newman, Private vs. common random bits in communication complexity, Information processing letters 39 (2) (1991) 67–71.

[11] Noam Nisan, Personal communication, 2011.

[12] Mihai Pătraşcu, Cc4: One-way communication and a puzzle 2009 (accessed January 20, 2011).

[13] Benny Porat, Ely Porat, Exact and approximate pattern matching in the streaming model, in: FOCS'09: Proc. 50th Annual Symp. Foundations of Computer Science, 2009, pp. 315–323.

[14] Andrew Chi-Chih Yao, Some complexity questions related to distributive computing, in: STOC'79: Proc. 11th Annual ACM Symp. Theory of Computing, 1979, pp. 209–213.