Electronic Colloquium on Computational Complexity, Report No. 128 (2012)

Streaming Complexity of Checking Priority Queues*

Nathanaël François[1] and Frédéric Magniez[2]

[1]Univ Paris Diderot, Sorbonne Paris-Cité, LIAFA, CNRS, 75205 Paris, France,
nathanael.francois@liafa.univ-paris-diderot.fr
[2]CNRS, LIAFA, Univ Paris Diderot, Sorbonne Paris-Cité, 75205 Paris, France,
frederic.magniez@univ-paris-diderot.fr

Abstract

This work is in the line of designing efficient checkers for testing the reliability of some massive data structures. Given a sequential access to the insert/extract operations on such a structure, one would like to decide, a posteriori only, if it corresponds to the evolution of a reliable structure. In a context of massive data, one would like to minimize both the amount of reliable memory of the checker and the number of passes on the sequence of operations.

Chu, Kannan and McGregor [9] initiated the study of checking priority queues in this setting. They showed that the use of timestamps allows to check a priority queue with a single pass and memory space $\tilde{O}(\sqrt{N})$. Later, Chakrabarti, Cormode, Kondapally and McGregor [7] removed the use of timestamps, and proved that more passes do not help.

We show that, even in the presence of timestamps, more passes do not help, solving an open problem of [9, 7]. On the other hand, we show that a second pass, but in *reverse* direction, shrinks the memory space to $\tilde{O}((\log N)^2)$, extending a phenomenon the first time observed by Magniez, Mathieu and Nayak [15] for checking well-parenthesized expressions.

1 Introduction

The reliability of memory is central and becomes challenging when it is massive. In the context of program checking [4] this problem has been addressed by Blum, Evans, Gemmell, Kannan and Naor [3]. They designed on-line checkers that use a small amount of reliable memory to test the behavior of some data structures. Checkers are allowed to be randomized and to err with small error probability. In that case the error probability is not over the inputs but over the random coins of the algorithm.

Chu, Kannan and McGregor [9] revisited this problem for priority queue data structures, where the checker only has to detect an error after processing an entire sequence of data accesses. This can be rephrased as a one-pass streaming recognition problem. Streaming algorithms sequentially scan the whole input piece by piece in one sequential pass, or in a small number of passes, while using sublinear memory space. In our context, the stream is defined by the sequence of insertions and extractions on the priority queue. Using a streaming algorithm, the objective is then to decide if the stream corresponds to a correct implementation of a priority queue. We also consider collection data structures that implement multisets.

*Supported by the French ANR Defis program under contract ANR-08-EMER-012 (QRAC project)

ISSN 1433-8092

Definition 1 (Collection,PQ)**.** *Let Σ_0 be some alphabet. Let $\Sigma = \{\mathit{ins}(a), \mathit{ext}(a) : a \in \Sigma_0\}$. For $w \in \Sigma^N$, define inductively multisets M_i by $M_0 = \emptyset$, $M_i = M_{i-1} \setminus \{a\}$ if $w[i] = \mathit{ext}(a)$, and $M_i = M_{i-1} \cup \{a\}$ if $w[i] = \mathit{ins}(a)$.*
Then $w \in$ Collection(Σ_0) if and only if $M_n = \emptyset$ and $a \in M_{i-1}$ when $w[i] = \mathit{ext}(a)$, for $i = 1, \dots, N$. Moreover, $w \in$ PQ(U), for $U \in \mathbb{N}$, if and only if $w \in$ Collection$(\{0, 1, \dots, U\})$ and $a = \max(M_{i-1})$ when $w[i] = \mathit{ext}(a)$, for $i = 1, \dots, N$.

Streaming algorithms were initially designed with a single pass: when a piece of the stream has been read, it is gone for ever. This makes those algorithms of practical interest for online context, such as network monitoring, for which first streaming algorithms were developed [1]. Motivated by the explosion in the size of the data that algorithms are called upon to process in everyday real-time applications, the area of streaming algorithms has experienced tremendous growth over the last decade in many applications. In particular, a streaming algorithm can model an external read-only memory. Examples of such applications occur in bioinformatics for genome decoding, or in Web databases for the search of documents. In that context, considering multi-pass streaming algorithm is relevant.

Using standard arguments one can establish that every p-pass randomized streaming algorithm needs memory space $\Omega(N/p)$ for recognizing Collection. Nonetheless, Chakrabarti, Cormode, Kondapally and McGregor [7] gave a one-pass randomized for PQ using memory space $\tilde{O}(\sqrt{N})$. They also showed that several passes does not help, since any p-pass randomized algorithm would require $\Omega(\sqrt{N}/p)$ passes. A similar lower bound was showed independently, but using different tools, by Jain and Nayak [10]. The case of a single pass was established previously by Magniez, Mathieu and Nayak [15] for checking the well-formedness of parenthesis expressions, or equivalently the behavior of a stack.

A simpler variant of PQ with timestamps was in fact first studied by Chu, Kannan and McGregor [9], where now each item is inserted to the queue with its index.

Definition 2 (PQ-TS)**.** *Let $\Sigma = \{\mathit{ins}(a), \mathit{ext}(a) : a \in \{0, 1, \dots, U\}\} \times \mathbb{N}$. Let $w \in \Sigma^N$. Then $w \in$ PQ-TS(U) if and only if $w \in$ Collection(Σ), $w[1, \dots, N][1] \in$ PQ(U), and $w[i][2] = i$ when $w[i][1] = \mathit{ins}(a)$.*

Nonetheless the two works [9, 7] let open two problems. The lower bound of [7] was only proved for PQ, and no significant lower bounds for PQ-TS was established. Moreover, the streaming complexity of PQ for algorithms that can process the stream in any direction has not been studied.

Even though recognizing PQ-TS is obviously easier than recognizing PQ, our first contribution (Section 3) consists in showing that they both obey the same limitation, even with multiple passes in the same direction.

Theorem 3. *Every p-pass randomized streaming algorithm recognizing* PQ-TS$(3N/2)$ *with bounded error $1/3$ requires memory space $\Omega(\sqrt{N}/p)$ for inputs of length N.*

As a consequence, since this lower bound uses very restricted hard instances, it models most of possible variations. For instance, assuming that the input is in Collection and has no duplicates, is not sufficient to guarantee a faster algorithm. The proof of Theorem 3 consists in introducing a related communication problem with $\Theta(\sqrt{N})$ players. Then we reduce the number of players to 3, and prove a lower bound on the information carried by players, leading to the desired lower bound. We are following the *information cost* approach taken in [8, 17, 2, 12, 11], among other works. Recently, the information cost appeared as one of the most central notion in communication complexity [6, 5, 13]. The information cost of a protocol is the amount of information that messages carry about players' inputs. We adapt this notion to suit both the nature of streaming algorithms and of our problem.

Even if our result suggests that allowing multiple passes does not help, one could also consider the case of bidirectional passes. We believe that it is a natural relaxation of multi-pass streaming algorithms where the stream models some external read-only memory. In that case, we show that a second pass, but in reverse order, makes the problem of checking PQ easy, even with no timestamps (Section 4). A similar phenomenon has been established previously in [15] for checking the well-formedness of parenthesis expressions. Their problem is simpler than ours, and therefore our algorithm is more general.

Theorem 4. *There is a bidirectional* 2*-pass randomized streaming algorithm recognizing* $\mathrm{PQ}(U)$ *with memory space* $\mathrm{O}((\log N)(\log U + \log N))$*, time per processing item* $\mathrm{polylog}(N, U)$*, and one-sided bounded error* N^{-c}*, for inputs of length* N *and any constant* $c > 0$*.*

Our algorithm uses a hierarchical data structure similar to the one introduced in [15] for checking well-parenthesized expressions. At high level, it also behaves similarly. It performs one pass in each direction and makes an on-line compression of past information in at most $\log N$ hashcodes. While this compression can lose information, the compression technique ensures that a mistake is always detected in one of the two directions. Nonetheless our algorithm differs on two main points. First, unlike parenthesized expressions, PQ is not symmetric. Therefore one has to design an algorithm for each pass. Second, the one-pass algorithm for PQ [7] is technically more advanced than the one of [15]. Thus designing a bidirectional 2-pass algorithm for PQ is more challenging.

Theorems 3 and 4 point out a strange situation but not isolated at all. Languages studied in [9, 15, 7, 14] and in this paper have space complexity $\Theta(\sqrt{N}\mathrm{polylog}(N))$ for a single pass, $\Omega(\sqrt{N}/p)$ for p passes in the same direction, and $\mathrm{polylog}(N)$ for 2 passes but one in each direction. We hope this paper makes progress in the study that phenomenon.

2 Preliminaries

In streaming algorithms (see [16] for an introduction), a *pass* on an input $w \in \Sigma^N$, for some alphabet Σ, means that w is given as an *input stream* $w[1], w[2], \dots, w[N]$, which arrives sequentially, i.e., letter by letter in this order. For simplicity, we assume throughout this article that the input length N is always given to the algorithm in advance. Nonetheless, all our algorithms can be adapted to the case in which N is unknown until the end of a pass.

Definition 5 (Streaming algorithm)**.** *A p-pass randomized* streaming algorithm *with space $s(N)$ and time $t(N)$ is a randomized algorithm that, given $w \in \Sigma^N$ as an input stream,*

- *performs k sequential passes on w;*
- *maintains a memory space of size at most $s(N)$ bits while reading w;*
- *has running time at most $t(N)$ per processed letter $w[i]$;*
- *has preprocessing and postprocessing time at most $t(N)$.*

The algorithm is bidirectional *if it is allowed to access to the input in the reverse order, after reaching the end of the input. Then p is the total number of passes in either direction.*

The proof of our lower bound uses the language of communication complexity with multi-players, and is based on information theory arguments. We consider *number-in-hand* and *message-passing* communication protocols. Each player is given some input, and can communicate with another player according to the rules of the protocol. Our players are embedded into a directed circle, so that each player can receive (resp. transmit) a message from its unique predecessor (resp. successor). Each player send a message after receiving one, until the end of the protocol is reached. Players have no space and time restriction. Only the number of rounds and the size of messages are constrained.

Consider a randomized multi-player communication protocol P. We consider only two types of random source, that we call *coins*. Each player has access to its own independent source of *private coins*. In addition, all players share another common source of *public coins*. The output of P is announced by the last player. This is therefore the last message of the last player. We say that P is with bounded error ϵ when P errs with probability at most ε over the private and public coins. The *transcript* Π of P is the concatenation of all messages sent by all players, including all public coins. In particular, it contains the output of P, since it is given by the last player. Given a subset S of players, we let Π_S be the concatenation of all messages sent by players in S, including again all public coins.

We now remind the usual notions of entropy H and mutual information I. Let X, Y, Z be random variables. Then $\mathrm{H}(X) = -\mathbb{E}_{x \leftarrow X} \log \Pr(X = x)$, $\mathrm{H}(X|Y = y) = -\mathbb{E}_{y \leftarrow Y} \log \Pr(X = x|Y = y)$, $\mathrm{H}(X|Y) = \mathbb{E}_{y \leftarrow Y} \mathrm{H}(X|Y = y)$, and $\mathrm{I}(X : Y|Z) = \mathrm{H}(X|Z) - \mathrm{H}(X|Y, Z)$. The entropy and the mutual information are non negative and satisfy $\mathrm{I}(X : Y|Z) = \mathrm{I}(Y : X|Z)$.

The mutual information between two random variables is connected to the Hellinger distance h between their respective distribution probabilities. Given a random variable X we also denote by X its underlying distribution.

Proposition 6 (Average encoding)**.** *Let X, Y be random variables. Then $\mathbb{E}_{y \leftarrow Y} \mathrm{h}^2(X|_{Y=y}, X) \leq \kappa \mathrm{I}(X : Y)$, where $\kappa = \frac{\ln 2}{2}$.*

The Hellinger distance also generalizes the cut-and-paste property of deterministic protocols to randomized ones.

Proposition 7 (Cut and paste)**.** *Let P be a 2-player randomized protocol. Let $\Pi(x, y)$ denote the random variable representing the transcript in P when Players A, B have resp. inputs x, y. Then $\mathrm{h}(\Pi(x, y), \Pi(u, v)) = \mathrm{h}(\Pi(x, v), \Pi(u, y))$, for all pairs (x, y) and (u, v).*

Last we use that the square of the Hellinger distance is convex, and the following connexion to the more convention ℓ_1-distance: $\mathrm{h}(X, Y)^2 \leq \frac{1}{2}\|X - Y\|_1 \leq \sqrt{2}\mathrm{h}(X, Y)$. For a reference on these results, see [10].

3 Lower bound for PQ-TS

The proof of our lower bound consists in first translating it into a $3m$-player communication problem, for some large m; then reducing the number of players to 3 using the information cost approach; and last studying the base case of 3 players using information theory arguments.

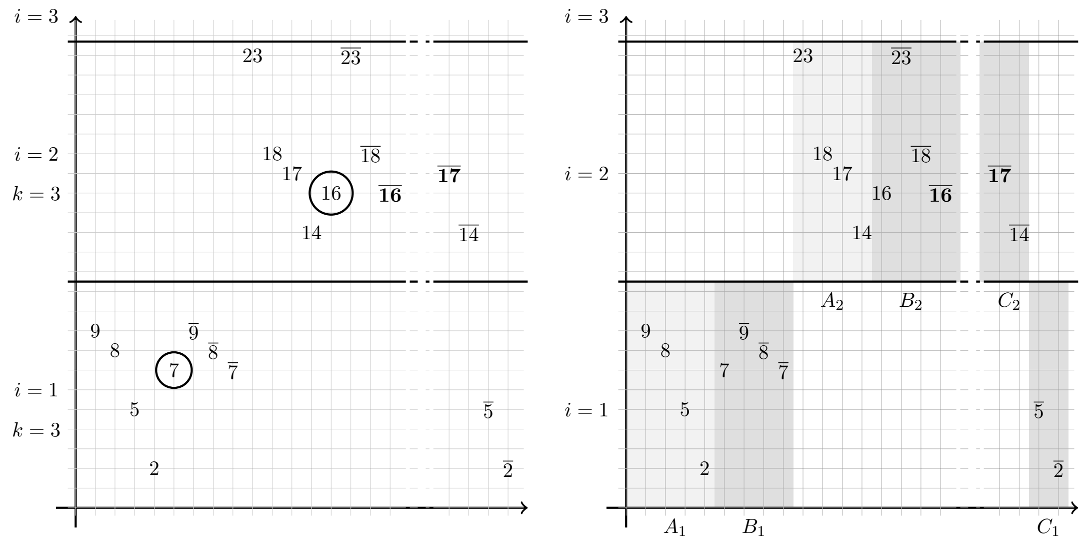

Figure 1: Left: Instance of RAINDROPS$(m, 4)$ with one error: 17 is extracted after 16. Insertions a_i are circled. Right: Cutting RAINDROPS$(m, 4)$ into $3m$ pieces to make it a communication problem. Players' input are within each corresponding region.

3.1 From streaming algorithms to communication protocols

In this section, we write a instead of `ins`(a) and $\bar{a}$ instead of `ext`(a). Consider the following set of hard instances of size $N = (2n + 2)m$:

RAINDROPS(m, n) (see LHS of Figure 1)

- For $i = 1, 2, \ldots, m$, repeat the following motif:
  - For $j = 1, 2, \ldots, n$, insert either $v_{i,j} = 3(ni - j)$ or $v_{i,j} = 3(ni - j) + 2$
  - Insert either $a_i = 3(ni-(k-1))+1$ or $a_i = 3(ni-k)+1$, for some $k \in \{2, \ldots, n\}$
  - Extract $v_{i,1}, v_{i,2}, \ldots, v_{i,k-1}, a_i$ in decreasing order
- Extract everything left in decreasing order

Observe that such an instance is in COLLECTION. One can also compute the timestamps of every element given its value. Last, there is only one potential error in each motif that can make it outside of PQ-TS. Indeed, $v_{i,1}, v_{i,2}, \ldots, v_{i,k-1}, a_i$ are in decreasing order up to eventually a switch between a_i and $v_{i,k-1}$.

Given such an instance as a stream, an algorithm for PQ-TS must decide if an error occurs between $\overline{a_i}$ and $\overline{v_{i,k}}$, for some i. Intuitively, if the memory space is less than εn, for a small enough constant $\varepsilon > 0$, then the algorithm cannot remember all the values $(v_{i,j})_j$ when a_i is extracted, and therefore cannot check a potential error with a_i. The next opportunity is during the last sequence of extractions. But then, the algorithm has to remember all values $(a_i)_i$, which is again impossible if the memory space is less than εm.

In order to formalize this intuition, Lemma 8 (proof in Appendix A) first translates our problem into a communication one between $3m$ players as shown on the RHS of Figure 1. Then we analyze its complexity using information theory arguments in Section 3.2.

Any insertion and extraction of an instance in RAINDROPS(m,n) can be described by its index and a single bit. Let $x_i[j] \in \{0,1\}$ such that $v_{i,j} = 3(ni-j) + 2x_i[j]$. Similarly, let $d \in \{0,1\}$ such that $a = 3(ni-k)+1+3d$. For simplicity, we write $\mathbf{x}$ instead of $(x_i)_{1 \leq i \leq m}$. Similarly, we use the notations $\mathbf{k}$ and $\mathbf{d}$. Then our related communication problem is:

WEAKINDEX(m,n)

- Input for players $(A_i, B_i, C_i)_{1 \leq i \leq m}$:
  - Player A_i has a sequence $x_i \in \{0,1\}^n$
  - Player B_i has $x_i[1, k_i - 1]$, with $k_i \in \{2,\dots,n\}$ and $d_i \in \{0,1\}$
  - Player C_i has $x_i[k_i, n]$
- Output: $f_m(\mathbf{x},\mathbf{k},\mathbf{d}) = \bigvee_{i=1}^m f(x_i,k_i,d_i)$, where $f(x,k,d) = [(d=0) \wedge (x[k]=1)]$
- Communication settings:
  - One round: each player sends a message to the next player according to the diagram $A_1 \to B_1 \to A_2 \to \cdots \to B_m \to C_m \to C_{m-1} \to \cdots \to C_1$.
  - Multiple rounds: If there is at least one round left, C_1 sends a message to A_1, and then players continue with the next round.

Lemma 8. *Assume there is a p-pass randomized streaming algorithm for deciding if an instance of* RAINDROPS(n,m) *is in* PQ-TS$(3mn)$ *with memory space $s(m,n)$ and bounded error ε. Then there is a p-round randomized protocol for* WEAKINDEX(n,m) *with bounded error ε such that each message has size at most $s(m,n)$.*

We are now ready to give the structure of the proof of Theorem 3, which has techniques based on information theory. Define the following collapsing distribution μ_0 of hard inputs (x,k,d), encoding instances of RAINDROPS$(1,n)$, where f always takes value 0. Distribution μ_0 is such that (x,k) is uniform on $\{0,1\}^n \times \{2,\dots,n\}$ and, given x,k, the bit $d \in \{0,1\}$ is uniform if $x[k]=0$, and $d=1$ if $x[k]=1$. From now on, (X,K,D) are random variables distributed according to μ_0, and (x,k,d) denote any of their values.

Then the proof of Theorem 3 consists in studying the information cost of any communication protocol for WEAKINDEX(n,m), which is a lower bound on its communication complexity. Using that μ_0 is collapsing for f, Lemma 9 establishes a direct sum on the information cost of WEAKINDEX(n,m). Then, even if f is constant on μ_0, Lemma 12 lower bounds the information cost of a single instance of WEAKINDEX$(n,1)$.

Proof of Theorem 3. Let n,N be positive integers such that $N=(2n+2)n$. Assume that there exists a p-pass randomized algorithm that recognizes PQ-TS$(3N/2)$, with memory space αn and bounded error ε, for inputs of size N. Then, by Lemma 8, there a p-round randomized protocol P for WEAKINDEX(n,n) such that each message has size at most αn. By Lemma 9, one can derive from P another $(p+1)$-round randomized protocol P' for WEAKINDEX$(n,1)$ with bounded error ε, and transcript Π' satisfying $|\Pi'| \leq 3(t+1)\alpha n$ and $\max\{\mathrm{I}(D:\Pi'_B|X,K), \mathrm{I}(K,D:\Pi'_C|X)\} \leq (p+1)\alpha$. Then by Lemma 12, $3(p+1)\alpha \geq (1-2\varepsilon)/10$, that is $\alpha = \mathrm{O}(1/p)$, concluding the proof. $\square$

3.2 Communication complexity lower bound

We first reduce the general problem $\textsc{WeakIndex}(n,m)$ with $3m$ players to a single instance of $\textsc{WeakIndex}(n,1)$ with 3 players. In order to do so we exploit the direct sum property of the information cost. The use of a collapsing distribution where f is always 0 is crucial.

Lemma 9. *If there is a p-round randomized protocol P for $\textsc{WeakIndex}(n,m)$ with bounded error ε and messages of size at most $s(m,n)$, then there is a $(p+1)$-round randomized protocol P' for $\textsc{WeakIndex}(n,1)$ with bounded error ϵ, and transcript P' satisfying $|\Pi'| \leq 3(p+1)s(m,n)$ and $\max\{\mathrm{I}(D:\Pi'_B|X,K), \mathrm{I}(K,D:\Pi'_C|X)\} \leq \frac{p+1}{m}s(m,n)$.*

Sketch of proof. Given a protocol P, we show how to construct another protocol P' for any instance (x,k,d) of $\textsc{WeakIndex}(n,1)$. In order to avoid any confusion, we denote by A, B and C the three players of P', and by $(A_i,B_i,C_i)_i$ the ones of P.

Protocol P'

- Using public coins, all players generate uniformly at random $j \in \{1,\dots,m\}$, and $x_i \in \{0,1\}^n$ for $i \neq j$
- Players A, B and C set respectively their inputs to the ones of A_j, B_j, C_j
- For all $i > j$, Player B generates, using its private coins, uniformly at random $k_i \in \{2,\dots,n\}$, and then it generates uniformly at random d_i such that $f(x_i,k_i,d_i)=0$
- For all $i < j$, Player C generates, using its private coins, uniformly at random $k_i \in \{2,\dots,n\}$, and then it generates uniformly at random d_i such that $f(x_i,k_i,d_i)=0$
- Players A, B and C run P as follows. A simulates A_j only, B simulates B_j and $(A_i,B_i,C_i)_{i>j}$, and C simulates C_j and $(A_i,B_i,C_i)_{i<j}$.

Observe that A starts the protocol if $j=1$, and C starts otherwise. Moreover C stops the simulation after p rounds if $j=1$, and after $p+1$ rounds otherwise. For all $i \neq j$, entries are generated such that $f(x_i,k_i,a_i)=0$, therefore $f_m(\mathbf{X},\mathbf{k},\mathbf{d}) = f(x_j,k_j,a_j) = f(x,k,a)$, and P' has the same bounded error than P.

Then we show in Appendix A that P' satisfies the required conditions of the lemma. $\square$

We now prove a trade-off between the bounded error of a protocol for a single instance of $\textsc{WeakIndex}(n,1)$ and its information cost. The proof involves some of the tools of [10] but with some additional obstacles to apply them. The inherent difficulty is due to that we have 3 players whereas the cute-and-paste property applies to 2-player protocols. Therefore we have to group 2 players together.

Given some parameters (x,k,a) for an input of $\textsc{WeakIndex}(n,1)$, we denote by $\Pi(x,k,a)$ the random variable describing the transcript Π of our protocol. We start by two lemmas exploiting the average encoding theorem (proofs in Appendix A).

Lemma 10. *Let P be a randomized protocol for $\textsc{WeakIndex}(n,1)$ with transcript Π satisfying $|\Pi| \leq \alpha n$ and $\mathrm{I}(K,D:\Pi_C|X) \leq \alpha$. Then*

$$\mathop{\mathbb{E}}_{x[1,l-1],l} \mathrm{h}^2(\Pi(x[1,l-1]0X[l+1,n],l,1), \Pi(x[1,l-1]1X[l+1,n],l,1)) \leq 28\alpha,$$

where $l \in [\frac{n}{2}+1,n]$ and $x[1,l-1]$ are uniformly distributed.

Lemma 11. *Let P be a randomized protocol for* $\textsc{WeakIndex}(n,1)$ *with transcript* Π *satisfying* $\mathrm{I}(D:\Pi_B|X,K)\leq\alpha$. *Then*

$$\mathop{\mathbb{E}}_{x[1,l-1],l}\mathrm{h}^2(\Pi(x[1,l-1]0X[l+1,n],l,0),\Pi(x[1,l-1]0X[l+1,n],l,1))\leq 12\alpha,$$

where $l\in[\frac{n}{2}+1,n]$ and $x[1,l-1]$ are uniformly distributed.

We now end with the main lemma which combines both previous ones and applies the cut-and-paste property, where Players A,C are grouped.

Lemma 12. *Let P be a randomized protocol for* $\textsc{WeakIndex}(n,1)$ *with bounded error ϵ, and transcript Π satisfying $|\Pi|\leq\alpha n$ and $\max\{I(D:\Pi_B|X,K),I(K,D:\Pi_C|X)\}\leq\alpha$. Then $\alpha\geq(1-2\varepsilon)/10$.*

Proof. Let L be a uniform integer random variable in $[\frac{n}{2}+1,n]$. Remind that we enforce the output of P to be part of Π. Therefore, any player, and in particular B, can compute f with bounded error ε given Π. Since $f(x[1,l-1]0X[l+1,n],l,0)=0$ and $f(x[1,l-1]1X[l+1,n],l,1)=1$, the error parameter ε must satisfies

$$\mathop{\mathbb{E}}_{x[1,l-1],l}\|\Pi(x[1,l-1]0X[l+1,n],l,0)-\Pi(x[1,l-1]1X[l+1,n],l,0)\|_1\geq 2(1-2\varepsilon).$$

The rest of the proof consists in upper bounding the LHS by 19α.

Applying the triangle inequality and that $(u+v)^2\leq 2(u^2+v^2)$ on the inequalities of Lemmas 10 and 11 gives

$$\mathop{\mathbb{E}}_{x[1,l-1],l}\mathrm{h}^2(\Pi(x[1,l-1]0X[l+1,n],l,0),\Pi(x[1,l-1]1X[l+1,n],l,1))\leq 30\alpha.$$

We then apply the cut-and-paste property by considering (A,C) as a single player with transcript $\Pi_{A,C}$. Therefore

$$\mathop{\mathbb{E}}_{x[1,l-1],l}\mathrm{h}^2(\Pi(x[1,l-1]0X[l+1,n],l,1),\Pi(x[1,l-1]1X[l+1,n],l,0))\leq 30\alpha.$$

Combining again with the inequality from Lemma 11 gives

$$\mathop{\mathbb{E}}_{x[1,l-1],l}\mathrm{h}^2(\Pi(x[1,l-1]0X[l+1,n],l,0),\Pi(x[1,l-1]1X[l+1,n],l,0))\leq 42\alpha.$$

Last, we get the requested upper bound by using the connexion between the Hellinger distance and the ℓ_1-distance, and the convexity of the square function. □

4 Bidirectional streaming algorithm for PQ

Remember that in this section our stream is given without any timestamps. Therefore we consider in this section only streams w of $\texttt{ins}(a)$, $\texttt{ext}(a)$, where $a\in[0,U]$. For the sake of clarity, we assume for now that the stream has no duplicate. Our algorithms can be extended to the general case, but the technical difficulties shadow the main ideas.

Up to padding we can assume that N is a power of 2: we append a sequence of $\texttt{ins}(a)\texttt{ext}(a)\texttt{ins}(a+1)\texttt{ext}(a+1)\ldots$ of suitable length, where a is large enough so that there is no duplicate (assuming that w is of even size, otherwise $w \notin \mathrm{PQ}(U)$). We use $\mathrm{O}(\log N)$ bits of memory to store, after the first pass, the number of letters padded.

We use a hash function based on the one used by the Karp-Rabin algorithm for pattern matching. For all this section, let p be a prime number in $\{\max(2U+1, N^{c+1}), \dots, 2\max(2U+1, N^{c+1})\}$, for some fixed constant $c \geq 1$. Since our hash function is linear we only define it for single insertion/extraction as

$$\mathrm{hash}(\mathtt{ins}(a)) = \alpha^a \mod p, \quad \text{and} \quad \mathrm{hash}(\mathtt{ext}(a)) = -\alpha^a \mod p,$$

where α is a randomly chosen integer in $[0, p-1]$. This is the unique source of randomness of our algorithm. A hashcode h *encodes* a sequence w if $h = \mathrm{hash}(w)$ as a formal polynomial in α. In that case we say that h *includes* $w[i]$, for all i. Moreover w is *balanced* if the same integers have been inserted and extracted. In that case it must be that $h = 0$. We also say that h is balanced it it encodes a balanced sequence w. The converse is also true with high probability by the Schwartz-Zippel lemma.

Fact 13. *Let w be some unbalanced sequence. Then* $\Pr(\mathrm{hash}(w) = 0) \leq \frac{N}{p} \leq \frac{1}{N^c}$.

The forward-pass algorithm was introduced in [7], but the reverse-pass one is even simpler. As a warming up, we start by introducing the later algorithm. In order to keep it simple to understand, we do not optimize it fully. Last define the instruction $\mathtt{Update}(h, v)$ that returns $(h + \mathrm{hash}(v) \mod p)$ *and* updates h to that value.

4.1 One-reverse-pass algorithm for PQ

Our algorithm decomposes the stream w into blocks. We call a valley an extraction $w[t] = \mathtt{ext}(a)$ with $w[t+1] = \mathtt{ins}(b)$. A new block starts at every valley. To the i-th block we associate a hashcode h_i and an integer m_i. Hashcode h_i encodes all the extractions within the block and the matching insertions. Integer m_i is the minimum of extractions in the block. With the values $(m_i)_i$, one can encode insertions in the correct h_i if $w \in \mathrm{PQ}$. Observe that we use index notations for block indices and bracket notations for stream positions.

Algorithm 1 uses memory space $\mathrm{O}(r)$, where r is the number of valleys in w. We could make it run with memory space $\mathrm{O}(\sqrt{N \log N})$ by reducing the number of valleys as in [7]. We do not need to as we use another compression in the two-pass algorithm.

We first state a crucial property of Algorithm 1, and then show that it satisfies Theorem 15, when there is no duplicate. We remind that we process the stream from right to left.

Lemma 14. *Consider Algorithm 1 right after processing $\mathtt{ins}(a)$. Assume that $\mathtt{ext}(a)$ has been already processed. Let $h_k, h_{k'}$ be the respective hashcodes including $\mathtt{ext}(a)$, $\mathtt{ins}(a)$. Then $k = k'$ if and only if all $\mathtt{ext}(b)$ occurring between $\mathtt{ext}(a)$ and $\mathtt{ins}(a)$ satisfy $b > a$.*

Theorem 15. *There is a 1-reverse-pass randomized streaming algorithm for $\mathrm{PQ}(U)$ with memory space $\mathrm{O}(r(\log N + \log U))$ and one-sided bounded error N^{-c}, for inputs of length N with r valleys, and any constant $c > 0$.*

Proof. We show that Algorithm 1 suits the conditions, assuming there is no duplicate. Let $w \in \mathrm{PQ}(U)$. Then w always passes the test at line 10. Moreover, by Lemma 14, each insertion $\mathtt{ins}(a)$ is necessarily in the same hashcode than its matching extraction $\mathtt{ext}(a)$. Therefore, all hashcodes

Algorithm 1: One-reverse-pass algorithm for PQ

```
1  $m_0 \leftarrow -\infty$; $h_0 \leftarrow 0$; $t \leftarrow N$; $i \leftarrow 0$ // $i$ is called the block index
2  While $t > 0$
3       If $w[t] = \texttt{ins}(a)$
4           $k \leftarrow \max\{j \leq i : m_j \leq a\}$; //Compute the hashcode index of $a$
5           Update$(h_k, w[t])$
6       Else $w[t] = \texttt{ext}(a)$
7           If $w[t+1] = \texttt{ins}(b)$ //This is a valley. We start a new block
8               $i \leftarrow i+1$; $m_i \leftarrow a$; $h_i \leftarrow 0$ //Create a new hashcode
9           Else $w[t+1] = \texttt{ext}(b)$
10              Check($a \geq b$) //Check that extractions are well-ordered
11          Update$(h_i, w[t])$
12      $t \leftarrow t-1$
13 For $j = 0$ to $i$: Check($h_j = 0$) //Check that hashcodes are balanced w.h.p.
14 Accept // $w$ succeeded to all checks
```

equal 0 at line 13 since they are balanced. In conclusion, the algorithm accepts w with probability 1.

Assume now that $w \notin$ PQ. First we show that unbalanced w are rejected with high probability, that is at least $1 - N^{-c}$, at line 13, if they are not rejected before. Indeed, since each $w[t]$ is encoded in some h_j, at least one h_j must be unbalanced. Then by Fact 13, the algorithm rejects w.h.p. We end the proof assuming w balanced. We remind that we process the stream from right to left. The two remaining possible errors are: (1) $\texttt{ins}(a)$ is processed before $\texttt{ext}(a)$, for some a; and (2) $\texttt{ext}(a), \texttt{ext}(b), \texttt{ins}(a)$ are processed in this order with $b < a$ and possibly intermediate insertions/extractions. In both cases, we show that some hashcodes are unbalanced at line 13, and therefore fail the test w.h.p by Fact 13, except if the algorithm rejects before.

Consider case (1). Since $\texttt{ins}(a)$ is processed before $\texttt{ext}(a)$, there is at least one valley between $\texttt{ins}(a)$ and $\texttt{ext}(a)$. Therefore $\texttt{ins}(a)$ and $\texttt{ext}(a)$ are encoded into two different hashcodes, that are unbalanced at line 13.

Consider now case (2). Lemma 14 gives that $\texttt{ext}(a)$ and $\texttt{ins}(a)$ are encoded in two different hashcodes, that are again unbalanced at line 13. □

4.2 Bidirectional two-pass algorithm

Algorithm 2 performs one pass in each direction using Algorithm 3. We use the hierarchical data structure of [15] in order to reduce the number of blocks. A block of size 2^i is of the form $[(q-1)2^i + 1, q2^i]$, for $1 \leq q \leq N/2^i$. Observe that, given two such blocks, either they are disjoint or one is included in the other. We decompose dynamically the letters of w, that have been already processed, into nested blocks of 2^i letters as follows. Each new processed letter of w defines a new block. When two blocks have same size, they merge. All processed blocks are pushed on a stack. Therefore, only the two topmost blocks of the stack may potentially merge. Because the size of each block is a power of 2 and at most two blocks have the same size (before merging), there are at most $\log N + 1$ blocks at any time.

Moreover, since our stream size is a power of 2, all blocks eventually appear in the hierarchical decomposition, whether we read the stream from left to right or from right to left. In fact, if two same-sized blocks appear simultaneously in one decomposition before merging, the same is true in

Algorithm 2: Bidirectional 2-pass algorithm for PQ

```
OnePassAlgorithm($w$)  reading stream from left to right
OnePassAlgorithm($w$)  reading stream from right to left
Accept   // $w$ succeeded to all checks
```

Algorithm 3: OnePassAlgorithm

```
$S \leftarrow []$;
If left-to-right-pass Then Push($S$,$(0,-\infty,0)$) // Initialization of $S$
While stream is not empty
    Read(next letter $v$ on stream) // See below
    While the 2 topmost elements of $S$ have same block size $\ell$
        $(h_1,m_1,\ell)$ ←Pop($S$); $(h_2,m_2,\ell)$ ←Pop($S$)
        Push($S$,$(h_1+h_2 \mod p, \min(m_1,m_2), 2\ell)$) // Merge of 2 blocks
If left-to-right-pass Then Check($S=[(0,-\infty,0),(0,0,N)]$)
Else Check($S=[(0,0,N)]$)}
Return

Function Read(v):
Case $v=\mathtt{ins}(a)$ // When reading an insertion
    Let $(h,m,\ell)$ be the first item of $S$ from top such that $a \geq m$
    Replace $(h,m,\ell)$ by $(\mathtt{Update}(h,v),m,\ell)$
    Push (S,$(0,+\infty,1)$)
Case $v=\mathtt{ext}(a)$ and left-to-right-pass // When reading an extraction
    For all items $(h,m,\ell)$ on $S$ such that $m>a$: Check($h=0$)
    Let $(h,m,\ell)$ be the first item of $S$ from top such that $a>m$
    Replace $(h,m,\ell)$ by $(\mathtt{Update}(h,v),m,\ell)$
    Push($S$,$(0,a,1)$)
Case $v=\mathtt{ext}(a)$ and right-to-left-pass // When reading an extraction
    For all items $(h,m,\ell)$ on $S$ such that $m>a$: Check($h=0$)
    Push($S$,$(\mathrm{hash}(v),a,1)$)
```

the other decomposition. This point is crucial for our analysis.

Algorithm 3 uses the following description of a block B: its hashcode h_B , the minimum m_B of its extractions, and its size ℓ_B. For the analysis, we also note t_B the index such that $w[t_B]=m_B$. Among those parameters, only h_B can change without B being merged with another block. On the pass from right to left, all extractions from the block and the matching insertions are included in h_B. On the pass from left to right, insertions are included in the hashcode of the earliest possible block where they could have been, and the extractions are included with their matching insertions. The minimums $(m_B)_B$ are used to decide where to include values (except extractions on the pass from right to left). Observe that it is important to check that $h_B=0$ whenever possible and not at the end of the execution of the algorithm, since only one block is left at the end.

When there is some ambiguity, we denote by $h^{\rightarrow}_B$ and $h^{\leftarrow}_B$ the hashcodes for the left-to-right and right-to-left passes. Observe that m_B, t_B, ℓ_B are identical in both directions.

Proof of Theorem 4. We show that Algorithm 2 suits the conditions, assuming there is no duplicate. The space constraints are satisfied because each element of S takes space $\mathrm{O}(\log N + \log U)$ and S has size at most $\log N + 1$. The processing time is from inspection.

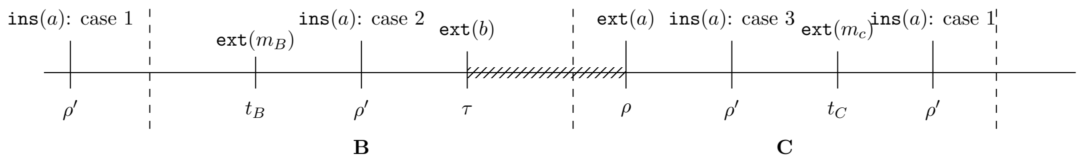

Figure 2: Relative positions of insertions and extractions used in the proof of Theorem 4

As with Theorem 15, inputs in PQ(U) are accepted with probability 1, and unbalanced inputs are rejected with high probability (at least $1-N^{-c}$). Let $w \notin$ PQ be balanced. For ease of notations, let $w[-1] = \texttt{ins}(-\infty)$ and $w[0] = \texttt{ext}(-\infty)$. Then, there are $\tau < \rho$ such that $w[\tau] = \texttt{ext}(b)$, $w[\rho] = \texttt{ext}(a)$, $a > b$, and $w[t] \neq \texttt{ins}(a)$ for all $\tau < t < \rho$.

Among those pairs (τ, ρ), consider the ones with the smallest ρ. From those, select the one with the smallest b, with $w[\tau] = \texttt{ext}(b)$. Let B, C be the largest possible disjoint blocks such that τ is in B and ρ in C. Then B and C have same size, are contiguous, and appear simultaneously in each direction before they merge. Let ρ' and τ' be such that $w[\rho'] = \texttt{ins}(a)$ and $w[\tau'] = \texttt{ins}(b)$. The minimality of ρ and the minimality of b guarantee that $w[t]$ is an insertion for all $\tau < t < \rho$. Indeed if $w[t] = \texttt{ext}(c)$ either $b > c$, which contradicts the minimality of b, or $c > b$ and (τ, t) contradicts the minimality of ρ. In particular, $t_C \geq \rho$ and $t_B \leq \tau$. Similarly $\tau < \tau'$, otherwise τ would be a better candidate than ρ.

We distinguish three cases based on the position ρ' of $\texttt{ins}(a)$ (see Figure 2): $\rho' \notin [t_B, t_C]$, $t_B < \rho' < \tau$, and $\rho < \rho' < t_C$. These cases determine in which hashcode $\texttt{ins}(a)$ is included. We analyze Algorithm 3 when some letter is processed before blocks potentially merge.

Case 1: $\rho' \notin [t_B, t_C]$. One can prove that $h^{\rightarrow}_B$ is unbalanced when $w[t_C]$ is processed and that $h^{\leftarrow}_C$ is unbalanced when $w[t_B]$ is processed; therefore Algorithm 3 detects w.h.p. $h^{\rightarrow}_B \neq 0$ or $h^{\leftarrow}_C \neq 0$ depending on whether $m_B > m_C$ (see Lemma 19 in Appendix B).

Case 2: $t_B < \rho' < \tau$. We show that when Algorithm 3 processes $w[t_B] = \texttt{ext}(m_B)$, it checks $h^{\leftarrow}_D = 0$ at line 23 for some $h^{\leftarrow}_D$ including $\texttt{ins}(a)$ but not $\texttt{ext}(a)$. Thus it rejects w.h.p.

When $w[\rho'] = \texttt{ins}(a)$ is processed on the right-to-left pass, $\tau \in B_1$ with B_1 a block in the stack. $\tau \in B$, therefore B_1 intersects B. Because $B_1 \not\subseteq B$, we have $B_1 \subseteq B$. Because $w[\tau] = \texttt{ext}(b)$, we have $a > b \geq m_{B_1}$, and block B_1 is eligible at line 14 of Algorithm 3, meaning that $w[\rho'] = \texttt{ins}(a)$ is included in either $h^{\leftarrow}_{B_1}$ or a more recent hashcode $h^{\leftarrow}_{B_2}$. Since $\rho' \in B$, again $B_2 \subseteq B$. Last, when Algorithm 3 processes $w[t_B] = \texttt{ext}(m_B)$, since we are still within B, some hashcode h_{B_3}, with $B_3 \subseteq B$, includes $w[\rho']$. Moreover, $h^{\leftarrow}_{B_3}$ does not include $w[\rho] = \texttt{ext}(a)$ since $\rho \in C$ and C comes before B. Last, $m_{B_3} > m_B$, by definition of m_B. Hence, Algorithm 3 checks $h^{\leftarrow}_{B_3} = 0$ at line 23 when processing $w[t_B]$. B_3 satisfies the conditions for D when $w[t_B]$ is processed, and Algorithm 3 rejects w.h.p.

Case 3: $\rho < \rho' < t_C$. The proof is the same as case 2, replacing τ, B, B_1, B_2, B_3, $h^{\leftarrow}_{B_1}$, $h^{\leftarrow}_{B_2}$, $h^{\leftarrow}_{B_3}$, t_B, C with ρ, C, C_1, C_2, C_3, $h^{\rightarrow}_{C_1}$, $h^{\rightarrow}_{C_2}$, $h^{\rightarrow}_{C_3}$, t_C, B and line 23 with line 18. Note that we only have $a \geq m_{C_1}$ this time, so it is important that the inequality at line 14 is large and not strict. $\square$

4.3 Generalization when duplicates occur

We maintain two additional parameters δ_B and C_B for each block B. The difference between the number of insertions and extractions included in h_B is stored in δ_B. Whenever $\delta_B = 0$, we check $h_B = 0$. The number of unmatched occurrences of $\texttt{ins}(m_B)$ for the left-to-right pass (resp. $\texttt{ext}(m_B)$ for the right-to-left pass) is stored in C_B. We can then appropriately determine whether each $\texttt{ext}(m_B)$ (resp. $\texttt{ins}(m_B)$) should be included in h_B.

The change on the criterion of line 14 of Algorithm 3 makes the proof of case 3 of the theorem longer and breaks the symmetry.

Acknowledgements

Authors would like to thank Rahul Jain and Ashwin Nayak for sharing their intuition and possible extensions of [10]. In particular, N.F. thanks Ashwin Nayak for having hosted him at IQC, University of Waterloo. They also thanks Christian Konrad, Wei Yu, Qin Zhang for related discussions, and Andrew McGregor for motivating us to study restricted instances of PQ.

References

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[2] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[3] M. Blum, W. S. Evans, P. Gemmell, S. Kannan, and M. Naor. Checking the correctness of memories. *Algorithmica*, 12(2):225–244, 1994.

[4] M. Blum and S. Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995.

[5] M. Braverman. Interactive information complexity. In *Proc. of ACM Symp. on Theory of Computing*, pages 505–524, 2012.

[6] M. Braverman and A. Rao. Information equals amortized communication. In 748-757, editor, *Proc. of IEEE Symp. on Foundations of Computer Science*, 2011.

[7] A. Chakrabarti, G. Cormode, R. Kondapally, and A. McGregor. Information cost tradeoffs for augmented index and streaming language recognition. In *Proc. of IEEE Symp. on Foundations of Computer Science*, pages 387–396, 2010.

[8] A. Chakrabarti, Y. Shi, A. Wirth, and A. C.-C. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Proc. of IEEE Symp. on Foundations of Computer Science*, pages 270–278, 2001.

[9] M. Chu, S. Kannan, and A. McGregor. Checking and spot-checking the correctness of priority queues. In *Proc. of Int. Colloquium on Automata, Languages and Programming*, pages 728–739, 2007.

[10] R. Jain and A. Nayak. The space complexity of recognizing well-parenthesized expressions in the streaming model: the index function revisited, 2010. ECCC Tech. Rep. TR10-071.

[11] R. Jain, J. Radhakrishnan, and P. Sen. A lower bound for the bounded round quantum communication complexity of Set Disjointness. In *Proc. of IEEE Symp. on Foundations of Computer Science*, pages 220–229, 2003.

[12] T. S. Jayram, Ravi Kumar, and D.Sivakumar. Two applications of information complexity. In *Proc. of ACM Symp. on Theory of Computing*, pages 673–682, 2003.

[13] I. Kerenidis, S. Laplante, V. Lerays, J. Roland, and D. Xiao. Lower bounds on information complexity via zero-communication protocols and applications. In *Proc. of IEEE Symp. on Foundations of Computer Science*, 2012. To appear.

[14] C. Konrad and F. Magniez. Validating XML documents in the streaming model with external memory. In *Proc. of Int. Conf. on Database Theory*, pages 34–45, 2012.

[15] F. Magniez, C. Mathieu, and A. Nayak. Recognizing well-parenthesized expressions in the streaming model. In *Proc. of ACM Symp. on Theory of Computing*, pages 261–270, 2010.

[16] S. Muthukrishnan. *Data Streams: Algorithms and Applications.* Now Publishers Inc., 2005.

[17] M. Saks and X. Sun. Space lower bounds for distance approximation in the data stream model. In *Proc. of ACM Symp. on Theory of Computing*, pages 360–369, 2002.

A Missing proofs for the lower bound

We start by proving the lemma relating the streaming complexity of deciding if an instance of RAINDROPS(m, n) belongs to PQ-TS$(3mn)$ to the communication complexity of WEAKINDEX(n, m).

Proof of Lemma 8. Assume that there exists a p-pass randomized streaming algorithm with memory space $s(m, n)$, that decides if an instance of RAINDROPS(m, n) belongs or not to PQ-TS$(3nm)$. Each instance of RAINDROPS(m, n) can be encoded by an input of WEAKINDEX(n, m), where each of the $3m$ players has one part of it. Then, the rest of the proof consists in showing how the players can use the algorithm in order to construct a protocol that satisfies the required properties of the lemma.

Each player simulates alternatively the algorithm. A player performs the simulation until the algorithm reaches the part of the input of the next player. Then the player sends the current state of the algorithm, so that the next player can continue the simulation. Since the algorithm uses at most memory space $s(m, n)$, the current state can be encoded using $s(m, n)$ bits. Each pass corresponds to one round of communication, implying the result. □

Before giving the next missing proofs of Section 3, we state some useful properties of entropy and mutual information that we need. See [10] for more information.

Fact 16. *Let X, Y, Z, R be random variables such X and Z are independent when conditioning on R, namely when conditioning on $R = r$, for each possible values of r. Then* $\mathrm{I}(X : Y|Z, R) \geq \mathrm{I}(X : Y|R)$.

Proof. From the definition of mutual information and the independence of X, Z when conditioning on R, we get that

$$\mathrm{I}(X : Y|Z, R) = \mathrm{H}(X|Z, R) - \mathrm{H}(X|Y, Z, R) = \mathrm{H}(X|R) - \mathrm{H}(X|Y, Z, R).$$

Using that entropy can only decrease under conditioning, and using again the definition of mutual information, we conclude by bounding the last term as

$$\mathrm{H}(X|R) - \mathrm{H}(X|Y, Z, R) \geq \mathrm{H}(X|R) - \mathrm{H}(X|Y, R) = \mathrm{I}(X : Y|R).$$

□

Proposition 17 (Chain rule). *Let X, Y, Z, R be random variables. Then $\mathrm{I}(X, Y : Z|R) = \mathrm{I}(X : Z|R) + \mathrm{I}(Y : Z|X, R)$.*

Proposition 18 (Data processing inequality). *Let X, Y, Z, R be random variables such that R is independent from X, Y, Z. Then $\mathrm{I}(X : Y|Z) \geq \mathrm{I}(f(X, R) : Y|Z)$, for every function f.*

Note that the previous property is usually stated with no variable T. Nonetheless, since T is independent from the other variables, we have $\mathrm{I}(X : Y|Z) = \mathrm{I}(X, R : Y|Z)$, and then we can apply the usual data processing inequality.

We can now prove our three lemmas.

End of proof of Lemma 9. Let Π, Π' be the respective transcripts of P, P'. For convenience, note $\Pi_{C_{m+1}} = \Pi_{B_m}$, $\Pi_{B_0} = \Pi_{C_m}$ and $\Pi_{C_{m+1}} = \Pi_{A_1}$. Remind that the public coins of a protocol are included in its transcript.

First, each player of P' sends 3 messages by round, and there are $(p + 1)$ rounds. Since each message has size at most $s(m, n)$, we derive that the length of Π' is at most $3(p + 1)s(m, n)$.

Then, in order to prove that there is only a small amount of information in the transcripts of Bob and Charlie, we show a direct sum of some appropriated notion of information cost. Consider first the transcript of Player C_1. Because of the restriction on the size of his messages, we know that $|\Pi_{C_1}| \leq (p + 1)s(m, n)$. From this we derive a first inequality on the amount of information this transcript can carry, using that the entropy of a variable is at most its bit-size:

$$\mathrm{I}(\mathbf{K}, \mathbf{D} : \Pi_{C_1}|\mathbf{X}) \leq |\Pi_{C_1}| \leq (p + 1)s(m, n).$$

We now use the chain rule in order to get a bound about the information carried by P' on a single instance.

$$\begin{aligned}
\mathrm{I}(\mathbf{K}, \mathbf{D} : \Pi_{C_1}|\mathbf{X}) &= \sum_{j=1}^{m} \mathrm{I}((K_i, D_i)_{j \geq i} : \Pi_{C_1}|\mathbf{X}, (K_i, D_i)_{i<j}) \quad \text{(by chain rule)} \\
&\geq \sum_{j=1}^{m} \mathrm{I}(K_j, D_j : \Pi_{B_{j-1}}|\mathbf{X}, (K_i, D_i)_{i<j}) \quad \text{(by data processing inequality)} \\
&\geq \sum_{j=1}^{m} \mathrm{I}(K_j, D_j : \Pi_{B_{j-1}}|\mathbf{X}) \quad \text{(by Fact 16)} \\
&= m \times \mathrm{I}(K_J, D_J : \Pi_{B_{J-1}}|\mathbf{X}, J) \quad \text{(by conditioning on } J\text{)} \\
&= m \times \mathrm{I}(K_J, D_J : \Pi_{B_{J-1}}, J, (X_i)_{i \neq J}|X_J) \text{ (independence of } J, (X_i)_{i \neq J}\text{)} \\
&= m \times \mathrm{I}(K, D : \Pi'_C|X) \quad \text{(since } J, (X_i)_{i \neq J} \text{ are public coins of } P'\text{)}.
\end{aligned}$$

We then do similarly for Player B_m and therefore conclude the proof. First the size bound on messages of B_m gives $\mathrm{I}(\Pi_{B_m} : \mathbf{D}|\mathbf{X}, \mathbf{K}) \leq (p+1)s(m,n)$. Then as before we get:

$$\begin{aligned}\mathrm{I}(\mathbf{D} : \Pi_{B_m}|\mathbf{X}, \mathbf{K}) &= \sum_{j=1}^{m} \mathrm{I}(D_j : \Pi_{B_m}|\mathbf{X}, \mathbf{K}, (D_i)_{i>j}) \geq \sum_{j=1}^{m} \mathrm{I}(D_j : \Pi_{C_{j+1}}|\mathbf{X}, \mathbf{K}, (D_i)_{i>j}) \\ &\geq m \times \mathrm{I}(D_J : \Pi_{C_{J+1}}, J, (X_i)_{i\neq J}|X_J, K_J) = m \times \mathrm{I}(D : \Pi'_B|X, K).\end{aligned}$$

□

Proof of Lemma 10. From the second hypothesis and the data processing inequality we get that $\mathrm{I}(K, D : \Pi_{A,C}|X) \leq \alpha$, which after applying the average encoding leads to $\mathbb{E}_{x,k,d}\,\mathrm{h}^2(\Pi_{A,C}(x,k,d), \Pi_{A,C}(x,K,D)) \leq \kappa\alpha$. We now restrict μ_0 by conditioning on $D = 1$. Then (X, K) is uniformly distributed. Moreover, since $D = 1$ with probability $3/4$ on μ_0, we get $\mathbb{E}_{x,k}\,\mathrm{h}^2(\Pi_{A,C}(x,k,1), \Pi_{A,C}(x,K,1)) \leq \frac{4}{3}\kappa\alpha$. Let J, L be uniform integer random variables respectively in $[2, \frac{n}{2}]$ and $[\frac{n}{2}+1, n]$. Then the above implies $\mathbb{E}_{x,j}\,\mathrm{h}^2(\Pi_{A,C}(x,j,1), \Pi_{A,C}(x,K,1)) \leq \frac{8}{3}\kappa\alpha$ and $\mathbb{E}_{x,l}\,\mathrm{h}^2(\Pi_{A,C}(x,l,1), \Pi_{A,C}(x,K,1)) \leq \frac{8}{3}\kappa\alpha$. Applying the triangle inequality and that $(u+v)^2 \leq 2(u^2+v^2)$, we get

$$\mathop{\mathbb{E}}_{x,j,l}\mathrm{h}^2(\Pi_{A,C}(x,j,1), \Pi_{A,C}(x,l,1)) \leq \tfrac{32}{3}\kappa\alpha.$$

Using the convexity of h^2, we finally obtain for $b = 0, 1$:

$$\mathop{\mathbb{E}}_{x[1,l-1],j,l}\mathrm{h}^2(\Pi_{A,C}(x[1,l-1]bX[l+1,n],j,1), \Pi_{A,C}(x[1,l-1]bX[l+1,n],l,1)) \leq \tfrac{64}{3}\kappa\alpha.$$

Now the chain rule allow us to measure the information about a single bit in $\Pi_{A,C}$ as

$$\begin{aligned}\mathrm{I}(X[L] : \Pi_{A,C}(X,J,1)|X[1,L-1]) &= \mathop{\mathbb{E}}_{l\leftarrow L}\mathrm{I}(X[l] : \Pi_{A,C}(X,J,1)|X[1,l-1]) \\ &= \frac{2}{n} \times \mathrm{I}(X[\tfrac{n}{2}+1,n] : \Pi_{A,C}(X,J,1)|X[1,\tfrac{n}{2}]).\end{aligned}$$

Since the entropy of a variable is at most its bit-size, we get that the last term is upper bounded by $|\Pi_{A,C}|$, which is at most αn by the first hypothesis. Then as before, the average encoding and the triangle inequality lead to

$$\mathop{\mathbb{E}}_{x[1,l-1],j,l}\mathrm{h}^2(\Pi_{A,C}(x[1,l-1]0X[l+1,n],j,1), \Pi_{A,C}(x[1,l-1]1X[l+1,n],j,1)) \leq 16\kappa\alpha.$$

Combining gives

$$\mathop{\mathbb{E}}_{x[1,l-1],l}\mathrm{h}^2(\Pi_{A,C}(x[1,l-1]0X[l+1,n],l,1), \Pi_{A,C}(x[1,l-1]1X[l+1,n],l,1)) \leq 28\alpha.$$

Let R_B be the random coins of B. Since they are independent from all variables, including the messages, the previous inequality is still true when we concatenate R_B to $\Pi_{A,C}$. Then Π_B is uniquely determined from R_B once $K, D, X[1, K-1]$ are fixed, which is the case in that inequality. Therefore replacing R_B by Π_B can only decrease the distance, concluding the proof. □

Proof of Lemma 11. Using the data processing inequality and the hypothesis we get that $\mathrm{I}(D : \Pi|X, K)) \leq \alpha$. Therefore by average encoding, $\mathbb{E}_{x,k,d}\, \mathrm{h}^2(\Pi(x, k, d), \Pi(x, k, D)) \leq \kappa\alpha$.

Let L be a uniform integer random variable in $[\frac{n}{2} + 1, n]$. Then $\mathbb{E}_{x,l,d}\, \mathrm{h}^2(\Pi(x, l, d), \Pi(x, l, D)) \leq 2\kappa\alpha$. Using the convexity of h^2 and the fact that $X[l]$ is a uniform random bit, we derive

$$\mathop{\mathbb{E}}_{x[1,l-1],l,d} \mathrm{h}^2(\Pi(x[1, l-1]0X[l+1, n], l, d), \Pi(x[1, l-1]0X[l+1, n], l, D)) \leq 4\kappa\alpha.$$

Since $D = 0$ with probability $1/2$ when $X[l] = 0$ and $K = l$, we finally get the two inequalities

$$\mathop{\mathbb{E}}_{x[1,l-1],l} \mathrm{h}^2(\Pi(x[1, l-1]0X[l+1, n], l, 0), \Pi(x[1, l-1]0X[l+1, n], l, D)) \leq 8\kappa\alpha,$$

$$\mathop{\mathbb{E}}_{x[1,l-1],l} \mathrm{h}^2(\Pi(x[1, l-1]0X[l+1, n], l, 1), \Pi(x[1, l-1]0X[l+1, n], l, D)) \leq 8\kappa\alpha,$$

leading to the conclusion using the triangle inequality and that $(u + v)^2 \leq 2(u^2 + v^2)$. □

B Missing proofs for the algorithm

We start by proving the property of Algorithm 1 we use in the proof of Theorem 15.

Proof of Lemma 14. Remind again, that we process the stream from right to left in this proof, and that $h_k, h_{k'}$ are the respective hashcodes including $\texttt{ext}(a), \texttt{ins}(a)$. First assume that all $\texttt{ext}(b)$ between $\texttt{ext}(a)$ and $\texttt{ins}(a)$ satisfy $b > a$. Let i be the current block index while processing $\texttt{ins}(a)$. Observe that k is the current block index right after processing $\texttt{ext}(a)$. Since $\texttt{ext}(a)$ is processed before $\texttt{ins}(a)$ and since there is a valley between $\texttt{ext}(a)$ and $\texttt{ins}(a)$, we have $k < i$.

We prove that $k' = \max\{j \leq i | m_j \leq a\} = k$. The first equality is from line 4 of Algorithm 1. We now prove the second equality. For each $j \in \{k+1, \ldots, i\}$, value m_j is extracted between $\texttt{ext}(a)$ and $\texttt{ins}(a)$. Then, our assumption leads to $m_j > a$. Moreover, because the algorithm checks at line 10 that extraction sequences included in the same hashcode are decreasing, we have $m_k \leq a$, leading to the second equality.

We now prove the converse by contrapositive. Assume that some $\texttt{ext}(b)$ between $\texttt{ext}(a)$ and $\texttt{ins}(a)$ satisfies $b \leq a$. Since we forbid duplicates, in fact $b < a$. Let j be the current block index right after processing $\texttt{ext}(b)$. Then line 10 ensures that $m_j \leq b$. Again, k is the current block index right after processing $\texttt{ext}(a)$, and therefore $k \leq j$. If $k = j$, then the extraction sequence is not decreasing and line 10 rejects, contradicting the hypotheses that the algorithm has not rejected yet after processing $\texttt{ins}(a)$. Therefore $k < j$. But, line 4 and the fact that $m_j \leq b$ imply that $k' \geq j$, and therefore $k < k'$. □

We now give the missing part of the proof of Theorem 4.

Lemma 19. *If $\rho' \notin [t_B, t_C]$, then Algorithm 2 rejects w with probability at least $1 - N^{-c}$.*

Proof. We prove that $h_B^{\rightarrow}$ is unbalanced when $w[t_C]$ is processed and that $h_C^{\leftarrow}$ is unbalanced when $w[t_B]$ is processed. From that, we deduce that the algorithm rejects with high probability unless $m_B \leq m_C$ and $m_C \leq m_B$, i.e. $m_B = m_C$, which is impossible because w has no duplicates and B and C are disjoint.

Indeed if $m_C < m_B$ then Algorithm 3 checks that $h_B^{\rightarrow} = 0$ at line 18 when processing $w[t_C]$, and rejects with high probability because $h_B^{\rightarrow}$ is unbalanced. Similarly, if $m_C < m_B$, it rejects with high probability at line 23 when processing $w[t_B]$ on the right-to-left pass.

Now we only have to prove that $h_B^{\rightarrow}$ (resp. $h_C^{\leftarrow}$) is unbalanced when $w[t_C]$ (resp. $w[t_B]$) is processed. Let us assume there exists $B_1 \subsetneq B$ such that $\texttt{ins}(a)$ is included in $h_{B_1}^{\rightarrow}$ when $w[t_B]$ is processed. Then, by definition of m_B, $m_{B_1} > m_B$. Moreover, $\rho \in C$, so $w[\rho] = \texttt{ext}(a)$ is not processed yet and not included in B_1. Therefore, Algorithm 3 checks $h_{B_1}^{\rightarrow} = 0$ at line 18, and rejects w.h.p. We can now assume that there is no such $B_1 \subsetneq B$, and therefore that h_B, does not include $\texttt{ins}(a)$ when $w[t_C]$ is processed. Since $h_B^{\rightarrow}$ includes $\texttt{ext}(a)$, $h_B^{\rightarrow}$ is unbalanced when t_C is processed.

The proof for $h_C^{\leftarrow}$ is the same as above, replacing $h_B^{\rightarrow}$, $h_{B_1}^{\rightarrow}$, B, B_1, t_B and t_C with $h_C^{\leftarrow}$, $h_{C_1}^{\leftarrow}$, C, C_1, t_C and t_B, and line 18 with line 23. $\square$