Streaming algorithms for the missing item finding problem

Manuel Stoeckl[*]

Abstract

Many problems on data streams have been studied at two extremes of difficulty: either allowing randomized algorithms, in the static setting (where they should err with bounded probability on the worst case stream); or when only deterministic and infallible algorithms are required. Some recent works have considered the adversarial setting, in which a randomized streaming algorithm must succeed even on data streams provided by an adaptive adversary that can see the intermediate outputs of the algorithm.

In order to better understand the differences between these models, we study a streaming task called "Missing Item Finding". In this problem, for $r < n$, one is given a data stream $a_1, \ldots, a_r$ of elements in $[n]$, (possibly with repetitions), and must output some $x \in [n]$ which does not equal any of the a_i. We prove that, for $r = n^{\Theta(1)}$ and $\delta = 1/\text{poly}(n)$, the space required for randomized algorithms that solve this problem in the static setting with error δ is $\Theta(\text{polylog}(n))$; for algorithms in the adversarial setting with error δ, $\Theta((1 + r^2/n)\text{polylog}(n))$; and for deterministic algorithms, $\Theta(r/\text{polylog}(n))$. Because our adversarially robust algorithm relies on free access to a string of $O(r \log n)$ random bits, we investigate a "random start" model of streaming algorithms where all random bits used are included in the space cost. Here we find a conditional lower bound on the space usage, which depends on the space that would be needed for a pseudo-deterministic algorithm to solve the problem. We also prove an $\Omega(r/\text{polylog}(n))$ lower bound for the space needed by a streaming algorithm with $< 1/2^{\text{polylog}(n)}$ error against "white-box" adversaries that can see the internal state of the algorithm, but not predict its future random decisions.

1 Introduction

A streaming algorithm is one which processes a long sequence of input data and performs a computation related to it. In general, we would like such algorithms to use as little memory as possible – preferably far less than the length of the input – while producing incorrect output with as low a probability as possible. For some problems, there is a space-efficient deterministic algorithm, which works for all possible inputs; but many others require randomized algorithms which, for any input, have a bounded probability of failure.

In the *adversarial setting*[4], one considers the case where a randomized algorithm is processing an input stream that is produced in real time, and furthermore the algorithm continually produces outputs depending on the partial stream that it has seen so far. It is possible that the outputs of the streaming algorithm will affect the future contents of the input stream; whether by accident or malice, this feedback may yield an input stream for which the randomized algorithm gives incorrect outputs. Thus, in the adversarial setting, we require that an algorithm has a bounded probability of failure, even when the input stream is produced by an adversary that can see all past outputs of the algorithm.

The extent to which an algorithm is vulnerable to adversaries depends critically on the use of randomness by the algorithm. If, given a randomized algorithm that has nonzero failure probability on any fixed input stream, an adversary somehow manages to determine all the past and future random choices made by an instance of the the algorithm, then the adversary can determine a specific continuation of the input stream on which the instance fails. Algorithms that are robust to adversaries often prevent the adversary from learning any of their important random decisions, and ensure that the decisions which are revealed do not affect the future performance of the algorithm. For example, [4] mentions a sketch-switching method in which a robust algorithm maintains multiple independent copies of a non-robust algorithm; it emits output derived from one non-robust instance until it reaches the point where an adversary might make the instance fail, at which point the algorithm switches to another instance, none of whose random choices have been revealed to the adversary yet.

Recent research has introduced models with requirements stronger than adversarial robustness. In the white-box streaming model[1], algorithms must avoid errors even when the adversary can see the current state of the

[*] Department of Computer Science, Dartmouth College.

This work was supported in part by the National Science Foundation under award 2006589.

algorithm (i.e, including past random decisions), but not future random decisions. In the pseudo-deterministic model[9], streaming algorithms should with high probability always give the same output for a given input; such algorithms are automatically robust against adversaries, because (assuming the algorithm has not failed) the outputs of the algorithm reveal nothing about any random decisions made by the algorithm.

In order to better understand the differences between all these models, we study a streaming problem known as Missing Item Finding (MIF). This problem is perhaps the simplest search problem for data streams where the space of possible answers shrinks as the stream progresses. For parameters $r < n$, given an data stream $a_1, \ldots, a_r$ of length r, where each element e_i is an integer in the range $[n]$, the goal of the $\textsc{mif}(n, r)$ problem is to identify some integer $x \in [n]$ for which, for all $i \in [r]$, $x \neq a_i$.

This problem is of interest because it has significantly different space complexities for regular randomized streaming algorithms, adversarially robust streaming algorithms, and deterministic streaming algorithms. Surprisingly, our adversarially robust algorithm when $r = \sqrt{n}$ needs oracle access to $\widetilde{O}(\sqrt{n})$ random bits, but only $\widetilde{O}(\log n)$ random bits of mutable memory. One of the main open problems left by our work is whether this is necessary. Our white-box model lower bound shows that the algorithm must make at least *some* random decisions that remain hidden from the adversary, and a conditional lower bound shows that, if the pseudo-deterministic space complexity of $\textsc{mif}(n, \sqrt{n})$ is $\widetilde{\Omega}(\sqrt{n})$, then the robust algorithm actually must use $\widetilde{\Omega}(n^{1/4})$ bits of space, including random bits.

1.1 Our results and contributions Our results – a series of upper and lower bounds for space complexities of $\textsc{mif}(n, r)$ in various different models are given in Table 1. For a more precise description of the models and of what guarantee exactly δ is associated with in each case, see Section 3.

Table 1: Table summarizing the upper and lower bounds on the space complexity of algorithms for $\textsc{mif}(n, r)$ in various models. δ is the worst case error – see Section 3 for what this means in the different models. $\star$: Unlike the other models, the complexity bounds for the zero error case are defined using of the *expected* algorithm space usage, not the worst-case space usage.

Model	Lower bound	Upper bound	Source
Classical	$\Omega(\sqrt{\frac{\log(1/\delta)}{\log(n)}} + \frac{\log(1/\delta)}{(\log n)(1+\log(n/r))})$ if $\delta \geq 1/n^r$	$\min(r, \frac{\log(1/\delta)}{\log(n/r)})$	Thm 4.1, Thm 4.2
Adv. Robust	$\Omega(\frac{r^2}{n} + \log(1-\delta))$	$O(\min(r, \left(1 + \frac{r^2}{n} + \ln\frac{1}{\delta}\right) \cdot \log r))$	Thm 5.1 , Thm 5.2
Zero error $\star$	$\Omega(\frac{r^2}{n})$	$O(\min(r, (1 + \frac{r^2}{n}) \log r)$	Thm 6.1, Thm 6.2
Deterministic	$\Omega(\sqrt{r} + \frac{r}{1+\log(n/r)})$	$O(\sqrt{r \log r} + \frac{r \log r}{\log n})$	Thm 7.1, Thm 7.2,
White box	$\Omega(r/(\log n)^4)$ if $\delta \leq 1/n^{O(\log n)}$	(see deterministic)	Thm 8.1
Random start	$\Omega(\sqrt{r}/\operatorname{polylog} n)$, assuming Pseudo-deterministic algs require $\Omega(r/\operatorname{polylog} n)$ bits	(see deterministic)	Thm 9.1

We shall highlight some of the more novel results in what follows:

- Our adversarially robust algorithm for $\textsc{mif}(n, r)$ uses its oracle-type access to random bits to keep track of a list L of outputs that it could give. At each point in time, Algorithm 5.1 outputs the first element of L which is still available. An adversary can choose to make the algorithm move to the next list element, but it cannot reliably provide an element from L that it has not yet seen. For the algorithm, switching to the next list element is easy – it just increments a counter – but keeping track of future intersections between the L and the stream requires that it record each intersecting element; fortunately, even with an adversary there will not be too many such intersections.

- Our deterministic algorithm for $\textsc{mif}(n, r)$ uses the (missing-) pigeonhole principle multiple times, and stays within a factor $\log r$ of the space lower bound. Algorithm 7.1 proceeds in several stages; in each stage, it considers a partition of the input space into a number of different parts, and maintains a bit vector keeping track of which part contains an element from the stream that arrived in the current stage. When there is exactly one part left, the algorithm remembers that part, discards the bit vector, and moves on to the next stage and a new partition of the input space. With suitably chosen partitions, the intersection of all the remembered parts from the different stages will be nonempty and disjoint from each element of the stream. The algorithm then reports an element from this intersection.

- Our white-box lower bound proof establishes an adversary that samples its next batch of inputs using a distribution ν over $[n]$ which is chosen so that the algorithm will also produce outputs distributed according to ν. This is done using recursive applications of Brouwer's fixed point theorem: for example, at the base level, we can use it because the map from the distribution on $[n]$ out of which the remaining input elements are sampled, to the distribution of the final algorithm output, is a continuous map from the space of distributions on $[n]$ to itself. Note that if ν picks some element with probability $\geq 2/3$, then the algorithm will also output that element with probability $\geq 2/3$, leading to a $\geq 1/3$ chance that the algorithm incorrectly emits an output that it received in the stream. We then show that, if a white box algorithm using less space than our lower bound exists, then said algorithm will fail with $\geq 1/2^{O((\log n)^2)}$ probability. This follows by an inductive argument which shows that, at any point in the stream, either the algorithm will make a mistake with significant probability, or there is a large enough chance that the next distribution which the adversary picks will be more "concentrated" than before, as measured by an ℓ_p norm for a value of p slightly larger than 1. As distributions cannot be infinitely "concentrated", it follows that the algorithm will eventually make a mistake with some low probability.

- Our conditional lower bound proof for the "random start" model, relies on the observation that at a given point in the stream, either the adversary is able to provide an input where it learns a lot about the initial random bits of the algorithm, or the algorithm, because it reveals very little about its internal randomness, also must consistently produce the same output at some point, in response to the same input. We can use this behavior to construct a pseudo-deterministic algorithm which works on a shorter input stream.

The rest of this paper is organized as follows. Related work is described in Section 2. Detailed descriptions of the models for streaming algorithms are given in Section 3. Sections 4 through 9 contain the main results of this paper, organized according to the rows of Table 1; they can be read in any order.

2 Related work

The Missing Item Finding problem appears to have been first studied by [20]. While they primarily consider the problem of finding a duplicate element in a stream of $m > n$ elements chosen from $[n]$, most of their results also apply to $\text{MIF}(n, n-1)$. For example, their multi-pass duplicate finding algorithms can easily be translated to multiple pass algorithms to find a missing element. Their main results also hold: they find an deterministic streaming algorithm for $\text{MIF}(n, n-1)$ using $O(\log n)$ bits of space must make $\Omega(\log n/\log\log n)$ passes over the stream, and claim that a single-pass deterministic algorithm for $\text{MIF}(n, n-1)$ requires at least $2^n - 1$ states.[1]

A variation on the Missing Item Finding problem, that forbids repeated elements in the input stream, was briefly studied in the first section of [18]. The paper mentions that for any $k \geq 1$, on a stream encoding a subset of $[n]$ of size $n-k$, it is possible to recover the remaining k elements with a sketch of size $O(k \log n)$. The paper [6] also briefly mentions a variant of Missing Item Finding to illustrate an exponential gap between space usage for regular randomized and adversarially robust streaming algorithms. For the problem where the stream can list any strict subset S of $[n]$, and one must recover a single element not in S, they observe that there is a randomized algorithm which uses an L_0-sampling sketch to solve the problem in $O((\log n)^2))$ space; but any adversarially robust algorithm that succeeds with high probability needs $\Omega(n)$ bits.

If we were to extend the Missing Item Finding problem to turnstile streams, then we would end up with something opposite to the "support-finding" streaming problem. In the support-finding problem, the algorithm is given a turnstile stream of updates to a vector $x \in \mathbb{Z}^{[n]}$; on querying the algorithm, it must return any index $i \in [n]$ where $x_i \neq 0$. [16] find that this problem – and the harder L_0 sampling problem, where one must find a

[1] As Algorithm 3.1 uses exactly 2^{n-1} states for $\text{MIF}(n, n-1)$, the value $2^n - 1$ may be a typo.

uniformly random element of the support of x – have a space lower bound of $\Omega\left(\min\left(n, \log\frac{1}{\delta}(\log\frac{n}{\log(1/\delta)})^2\right)\right)$. This is close to [14]'s L_0 sampling algorithm which uses $O(\log\frac{1}{\delta}(\log n)^2)$ bits of space.

The paper [17] studies a two player game that is similar to Missing Item Finding. Here there are two players, a ”Dealer” and a ”Guesser”: for each of n turns, the players simultaneously do the following: the Dealer chooses a number from $[n]$ that it has not picked so far, and the Guesser guesses a number in $[n]$. The goal of the Guesser is to maximize expected score, the number of times their number matches the Dealer's choice; the Dealer tries to minimize the score. The paper proves upper and lower bounds on the expected score, for a number of scenarios. Notably, a Guesser that is limited to remember only m bits of information can do much better against a static Dealer (that chooses a hard ordering of numbers at the start of the game) than against an adaptive Dealer (that may choose the next number depending on the guesses made by the Guesser.) For example, $m = O((\log n)^2)$ suffices for an expected score of $\Omega(\log n)$ against a static Dealer, but there exists an adaptive Dealer which limits any Guesser's expected score to $(1+o(1))\ln m+O(\log\log n)$. The objectives of the Guesser and Dealer are similar to those of the algorithm and adversary in Missing Item Finding: the Guesser tries to avoid, if possible, guessing any value that the Dealer has revealed before; while the Dealer tries to ensure the Guesser chooses that the Dealer had already sent before. However, unlike Missing Item Finding, the Dealer-Guesser game requires that numbers dealt never be repeated and that all numbers be used, which makes it much easier to identify a number that will be dealt in the future.

In the Mirror Game of [8], there are two players, Alice and Bob who alternately declare numbers from the set $[2n]$. The players lose if they declare a number that has been declared before. Since Alice goes first, even if Bob can only remember $O(\log n)$ bits about the history of the game, Bob still has a simple strategy that will not lose. On the other hand, [8] prove that in order for Alice to guarantee a draw against Bob, they require $\Omega(n)$ bits of memory. If a low probability of error is acceptable, [7] provide a randomized strategy for Alice with $O((\log n)^3)$ bits of memory that draws with high probability – but this requires oracle access to a large number of random bits, or cryptographic assumptions. [7] and [17] ask whether there is a strategy using $O(\operatorname{polylog} n)$ bits of memory and of randomness. (Again, the objective of Alice in this game is quite similar to that of the algorithm in Missing Item Finding – but numbers are never repeated, and all numbers in $[2n]$ are used by the end of the game.)

The problem of constructing an adversarially resilient Bloom filter is addressed by [19]. Here one seeks a an ”approximate set membership” data structure, which is initialized on a set S of size n, and thereafter answers queries of the form ”is $x \in S$” with false positive error probability ϵ. An implementation of this structure is adversarially resilient if the false positive probability of the last element in the sequence is still $\leq \epsilon$ when the adversary chooses the sets S, and adaptively chooses the sequence of t elements to query. In addition to lower and upper bound results conditional on the existence of one-way functions, [19] find a construction for an adversarially resilient bloom filter using $O(n\log 1/\epsilon + t)$ bits of memory.

There are many papers on the topic of adversarially robust streaming. Among them, we mention [11], who prove that linear sketches on turnstile streams are not, in general, robust against adversaries. [5] find that algorithms based on finding a representative random sample of the elements in a stream may need only slight modification to work with adaptive adversaries; [4] establish general methods to convert streaming algorithms with real valued output that are not robust against adversaries to ones which are, in exchange for an increase in space usage. [12] improve on the space tradeoff of this result by using differential privacy. [21] improve the space/approximation factor tradeoffs for adversarially robust algorithms on tasks like F_p estimation.

The thread of finding separations between the space needed for classical streaming and for adversarially robust streaming has been pursued by [15], who construct a problem whose classical and adversarially robust space complexities are exponentially separated. [6] mention that this also holds for the variant of Missing Item Finding mentioned above, and prove a separation for the adversarially robust space complexity of graph coloring on insertion streams.

Pseudo-deterministic streaming algorithms were first studied by [9]; the paper finds a separation between the classical and pseudo-deterministic memory needed for the task of finding a nonzero entry of a vector given by turnstile updates from a stream, among other problems. While it is not a streaming task, the Find1 query problem – in which one is given a bit vector x with $\geq 1/2$ density of ones, and must find an index i where $x_i = 1$ by querying coordinates – has been found to require significantly more queries in the pseudo-deterministic case than in the general randomized case [10].

Streaming algorithms robust against white box adversaries were considered by [1]; they rule out efficient white-box adversarially robust algorithms for tasks like F_p moment estimation, while finding algorithms for heavyhitters-type problems. They also show how to reduce white-box adversarially robust algorithms to deterministic 2-party communication protocols, where lower bounds may be easier to prove.[2]

The Missing Item Finding problem has connections to graph streaming problems. Just as the L_0-sampling problem has been used by streaming algorithms that find a structure in a graph, behaviors like those of the Missing Item Finding problem appear in algorithms that look for a structure which is not in a graph. Specifically, the graph coloring problem is equivalent to finding a small collection of cliques which cover all vertices but do not include any edge in the graph. [3] proved that general randomized streaming algorithms can $\Delta + 1$ color a graph in $\widetilde{O}(n)$ space, where n is the number of vertices. [6] showed that adversarially robust streaming algorithms in $\widetilde{O}(n)$ space must use at least Δ^2 colors for a graph of maximum degree Δ; and [2] proved that deterministic streaming algorithms using $\widetilde{O}(n)$ space must use $\exp(\Delta^{\Omega(1)})$ colors. The papers [6] and [2] are noteworthy in particular because their lower bound proofs use essentially the same arguments as this paper's lower bound proofs for Missing Item Finding. (In fact, our proof of Theorem 5.1 was inspired by the [6]'s proof, while Theorem 7.1 was independently developed.) Because of this, we suspect that this paper's white box lower bound will have an analogue for graph coloring.

3 Preliminaries

Notation In this paper, following standard convention, $[n]$ is the set $\{1, 2, \ldots, n\}$, and $\binom{X}{k}$ is shorthand for the set of all subset of X of size k. For a finite set Y, we let $\triangle Y$ be the set of all probability distributions over Y. For π a probability distribution over Y, we write $\alpha \sim \pi^k$ to mean that $\alpha \in Y^k$ and each coordinate of α is chosen independently at random according to π. For some $x \in Y$, the distribution $\mathbb{1}_y$ is value 1 on y and value 0 everywhere else; drawing a sample from this distribution will always result in y. The p-norm of a distribution ϕ on Y is written as $\|\phi\|_p := \left(\sum_{i \in Y} \phi(i)^p\right)^{1/p}$. The notation $[t]^\star$ gives the set of all sequences of elements from t, of any length. The empty sequence is written ϵ; a sequence $s \in [t]^\star$ may be written as $(s_1, s_2, \ldots, s_k)$, in which case its length $|s| = k$. To concatenate two sequences a and b, we write "$a.b$". $\widetilde{O}(x)$ means $O(x \operatorname{polylog}(x))$, and $\widetilde{\Omega}(x)$ means $\Omega(x/\operatorname{polylog}(x))$,

A simple algorithm While in most cases there are more efficient alternatives, this algorithm for $\text{MIF}(n, r)$ is particularly simple:

Algorithm 3.1. A simple deterministic streaming algorithm for $\text{MIF}(n, r)$

```
Initialization:
$x \leftarrow \{0, \ldots, 0\}$, a vector in $\{0, 1\}^{[r]}$

Update($e \in [n]$):
if $e \le r$ then
    $x_e \leftarrow 1$
end if

Query:
if $\exists j \in [r] : x_j = 0$ then
    output: $j$
else
    output: $r + 1$
end if
```

3.1 Models for streaming algorithms We now precisely define the models of streaming computation considered in this paper. We classify the models by the type of randomness used, the measure of the cost of the algorithm, the setting in which they are measured, and by any additional constraints.

Randomness A streaming algorithm for $\text{MIF}(n, r)$ has a set Σ of possible states; a possibly random initial state $s_{\text{init}} \in \Sigma$, a possibly random transition function $\tau : \Sigma \times [n] \to \Sigma$, and a possibly random output function

[2] Unfortunately, for Missing Item Finding, the natural 2-party communication game is $\text{AVOID}(n, r/2, r/2)$, whose deterministic communication lower bound is almost the same as the randomized lower bound. See Section 3.2. In contrast, our deterministic and white box lower bounds both use $O(\log n)$ players/adaptive steps.

$\omega : \Sigma \to [n]$. The models of this paper will use the following four variations:

1. **Random oracle**: The initial state, transition function, and output function may all be random and correlated; i.e, there is a space Ω and random variable R on that space for which s_{init} is a function of R, and $\tau(s, a) = f(s, a, R)$ for some deterministic function $f : \Sigma \times [n] \times \Omega \to \Sigma$, and $\omega(s) = g(s, R)$ for some deterministic function $g : \Sigma \times \Omega \to [n]$. We can view this as the algorithm having access to an oracle for all of its operations, which provides the value of the variable R.

2. **Random tape**: In this case, the initial state, transition function, and output function are all random, but they are uncorrelated; each step i of the algorithm has associated random variables $R_{i,\tau}$ and $R_{i,\omega}$, and all of these variables are independent of each other and of the initial state s_{init}. The transition function of the algorithm is $\tau(s, a) = f(s, a, R_{i,\tau})$ for some f, and the output function is $\omega(s) = g(s, R_{i,\omega})$ for some g. If the algorithm visits a state twice, the transitions and outputs from that state will be independent. Intuitively, with this type of access to randomness, the algorithm can always sample fresh random bits (i.e, reading forward on a tape full of random bits), but cannot remember them for free.

3. **Random seed**: Here the initial state s_{init} may be chosen randomly, but the transition function and output function are deterministic. The algorithm only has access to the randomness it had when it started.

4. **Deterministic**: The initial state is fixed, and the transition function and output function are deterministic.

These variations are listed in decreasing order of strength; the random oracle model can emulate the random tape model, which is stronger than the random seed model, which is stronger than the deterministic model. Note that the random oracle model, while inconvenient to implement exactly due to the need to store all the random bits used, can be approximated in practice, since a cryptographically secure random number generator can be used to generate all the random bits from a small random seed.[3] Of course, if modern CSPRNGs based on functions like AES are broken, or one-way functions are proven not to exist, then the random oracle model may prove unreasonable.

Cost measure In this paper, the space cost of an algorithm is the worst case value, over all possible streams or adversaries, of either the *maximum* number of bits used by the algorithm, or the *expected* number of bits used by the algorithm. The number of bits required is determined by a prefix-free encoding of the set Σ of states as strings in $\{0, 1\}^\star$; for most models, we measure the maximum number of bits used, which is $\lceil \log |\Sigma| \rceil$ for the best encoding.

Setting The cost of an algorithm, and its probability of an error, are measured against the type of inputs that it is given.

1. **Static**: In the static setting, the algorithm should give an incorrect output, on being queried at the end of the stream, with probability $\leq \delta$, when it is given any fixed input stream.[4]

2. **Adversarial**: In the adversarial setting, we consider the algorithm as being part of a two player game between it and an adversary; the algorithm receives a sequence of elements $e_1, \ldots, e_r$ from the adversary, and after each element e_i, the algorithm shall produce an output o_i corresponding to the sequence $e_1, \ldots, e_i$. The adversary chooses input e_i based on the transcript $o_0, e_1, o_1, e_2, \ldots, o_{i-1}$ that has been seen so far. The probability that the sequence of outputs produced by the algorithm has an error should be be $\leq \delta$, for any adversary.

3. **White box adversarial**: This is similar to the adversarial setting, except that here the adversary chooses the next input e_i as a function of the current state s_i of the algorithm. Here, the probability that the algorithm should make a mistake when producing an output at the end of the stream should be $\leq \delta$.

[3] As the space cost of this seed can be shared between all tasks performed by a computer, we do not account for it in the space cost estimates for this paper.

[4] This is a weaker condition than requiring that the entire sequence of intermediate outputs of the algorithm is correct; however, our lower bounds in static and white-box adversarial settings only require this weaker condition.

Extra constraints A streaming algorithm may be required to be pseudo-deterministic; in other words, for any input stream $\sigma = e_1, \ldots, e_r$, there should be a corresponding output o_σ of the algorithm for which the algorithm is considered to have made a mistake if it does not output o_σ. In other words, the algorithm should (with probability $\geq 1 - \delta$) behave as if it were deterministic.

A noteworthy constraint which we do not consider in the following set of models, is the requirement that the algorithm detects when its next output is not certain to be correct, and if so, aborts instead of producing the wrong value. Most of the algorithms presented in this paper already have this property – the one exception, Algorithm 4.1, can be patched to do so at the cost of an extra bit of space.

Models The models of this paper are described by the following table:

Model	Setting	Randomness	Cost	Extra conditions
Classical	Static	Oracle	Maximum space	
Robust	Adversarial	Oracle	Maximum space	
Zero error	Static	Oracle	Expected space	$\delta = 0$
Deterministic	Static	Deterministic	Maximum space	
White box robust	White-box adv.	Tape	Maximum space	
Pseudo-deterministic	Static	Oracle	Maximum space	Pseudo-deterministic
Random start	Adversarial	Seed	Maximum space	

A brief note on the "Zero error" model; this is a special case where the algorithm may be randomized, but is required to always give correct output for any input stream; unlike the deterministic model, the cost of the algorithm is the *expected* number of bits of space used by the algorithm. We include this model because, in many cases, a computer may run many independent instances of a streaming algorithm, and it is often more important that the instances do not fail than that they hold to strict space limits. In this scenario, as long as the expected space used by each algorithm is limited, and the worst case space usage is not too extreme, by the Chernoff bound it is unlikely that the total space used by all the instances exceeds the expected space by a significant amount. Unlike the case for time complexity, where a Las-Vegas algorithm can be obtained by repeating a Monte-Carlo algorithm until the solution is verifiably correct, there is no simple way to construct a single-pass, zero-error streaming algorithm from one with nonzero error.

We use the following notation for the space complexities of these models. The δ-error space complexity of the classical model for a task T is $S_\delta(T)$; for the robust model, $S_\delta^{AR}(T)$, for the zero error model, $S_0(T)$; for the deterministic model, $S^{det}(T)$; for the white box robust model, $S_\delta^{WB}(T)$; for the pseudo-deterministic model, $S_\delta^{PD}(T)$, and the random start model, $S_\delta^{RS}(T)$. The following relationships follow from the definitions of the models:

$$\begin{array}{ccc} & S_\delta^{AR}(T) \leq S_\delta^{RS}(T) & S_\delta^{RS}(T) \leq S^{det}(T) \\ S_\delta(T) \leq S_\delta^{AR}(T) & S_\delta^{AR}(T) \leq S_\delta^{WB}(T) & S_\delta^{WB}(T) \leq S^{det}(T) \\ & S_\delta^{AR}(T) \leq S_\delta^{PD}(T) & S_\delta^{PD}(T) \leq S^{det}(T) \\ & & S_0(T) \leq S^{det}(T) \end{array}$$

For problems in communication complexity, we write $R_\delta^{\rightarrow}(T)$ for the one-way randomized δ-error communication complexity of task T, and $D^{\rightarrow}(T)$ for the deterministic communication complexity.

3.2 Lemmas

The avoid(t, a, b) communication task This one-way communication game was introduced by [6]. In it, Alice is given $S \subseteq [t]$ with $|S| = a$, and sends a message to Bob, who must produce $T \subseteq [t]$ with $|T| = b$ where T is disjoint from S.

Lemma 3.1. *(From [6], Lemma 6) The public-coin δ error one-way communication complexity of* avoid(t, a, b) *is at least* $\log(1 - \delta) + \log(\binom{t}{a} / \binom{t-b}{a})$. *Because*

$$\binom{t}{a} \Big/ \binom{t-b}{a} = \frac{t!(t-a-b)!}{(t-a)!(t-b)!} \geq 2^{\frac{ab}{t \ln 2}}$$

we have the weaker but more convenient lower bound $R_\delta^{\to}(\textsc{avoid}(t,a,b)) \geq \frac{ab}{t\ln 2} + \log(1-\delta)$

The above lower bound is mainly useful when $ab \in [t, t^2]$. For smaller inputs:

Lemma 3.2. *The public-coin δ-error one-way communication complexity of* $\textsc{avoid}(t,a,b)$ *satisfies*

$$R_\delta^{\to}(\textsc{avoid}(t,a,b)) \geq \min\left(\log(a+1), \log\frac{\ln(1/\delta)}{\ln(et/a)}\right) .$$

For the deterministic case, we have $D^{\to}(\textsc{avoid}(t,a,b)) \geq \log(a+1)$.

Proof. Say we have a public coin one-way randomized protocol Π for $\textsc{avoid}(t,a,b)$ with error δ; by the averaging argument, there exists a fixing of the randomness of the protocol, which is correct on $\geq 1-\delta$ of the sets in $\binom{[t]}{a}$. Let Ψ be this deterministic protocol, and let $\hat{m}$ be the number of distinct messages sent by Ψ. Each message $i \in [\hat{m}]$ corresponds to some set B_i that Bob outputs on receiving the message. Let $E := \{e_1, \ldots, e_m\}$ be a hitting set for $\{B_i\}_{i\in[\hat{m}]}$ of size $m \leq \hat{m}$; i.e, for all B_i, there is some $e_j \in B_i$. Let $\mathcal{C} \subseteq \binom{[t]}{a}$ be the set of inputs for which Ψ is correct; we note that no inputs in $\mathcal{C}$ can contain all of E, because if $A \supseteq E$, then every B_i intersects A, making the protocol fail. Assuming $m \leq a$, we have:

$$\begin{aligned}\delta \geq 1 - |\mathcal{C}|/\binom{t}{a} &\geq |\{A \in \binom{t}{a} : A \supseteq E\}|/\binom{t}{a} \\ = \binom{t-m}{a-m}/\binom{t}{a} &= \frac{a\cdot(a-1)\cdots(a-m+1)}{t\cdot(t-1)\cdots(t-m+1)} \geq \left(\frac{a/e}{t}\right)^m ,\end{aligned}$$

where the last step is derived from the well known inequality $a! \geq (a/e)^a$. Rearranging gives $m \geq \ln(1/\delta)/\ln(et/a))$. In the case where $m > a$, this argument does not work, because then $\binom{t-m}{a-m} = 0$. Combining the two cases gives: $\hat{m} \geq m \geq \min(a+1, \ln(1/\delta)/\ln(et/a)))$. Thus $R_\delta^{\to}(\textsc{avoid}(t,a,b)) \geq \log(\min(a+1, \ln(1/\delta)/\ln(et/a)))$.

For general deterministic protocols, we reuse the analysis of randomized protocols with $\delta = 0$, concluding that $D^{\to}(\textsc{avoid}(t,a,b)) \geq \log(a+1)$. □

The following lemma is a simple variation of Chernoff's and Azuma's inequalities; for completeness, we present a proof in Appendix A.

Lemma 3.3. (Modified Azuma's inequality) *Let* $X_1, \ldots, X_n$ *be* $\{0,1\}$ *random variables, with* $\mathbb{E}[X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq p$ *for all* i *and all* $x_1, \ldots, x_n \in \{0,1\}^n$*. Then*

$$\Pr\left[\sum_{i=1}^n X_i \geq np(1+\delta)\right] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{np} \leq e^{-\frac{\delta^2}{2+\delta}np} .$$

A number of versions of Brouwer's fixed point theorem have been proven; in this paper, we will use the following, which is equivalent to Corollary 2.15 of [13].

Lemma 3.4. (Brouwer's fixed point theorem) *Every continuous map from a space homeomorphic to an n dimensional-ball to itself has a fixed point.*

4 Classical model

Theorem 4.1. *For any* $\delta \leq 1/(2n)$*, the space complexity for an algorithm solving* $\textsc{mif}(n,r)$ *with error* $\leq \delta$ *is* $S_\delta(\textsc{mif}(n,r)) \geq S^{\text{DET}}(\textsc{mif}(\lceil \frac{n}{t} \rceil, \lfloor \frac{r}{t} \rfloor))$*, for* $t = \left\lceil \frac{r\log n}{\log\frac{1}{2\delta}} \right\rceil$*. If we apply the upcoming lower bound from Theorem 7.1 on the deterministic space complexity of* $\textsc{mif}$*, we get:*

$$S_\delta(\textsc{mif}(n,r)) \geq \Omega\left(\sqrt{\min\left(r, \frac{\log(1/\delta)}{\log n}\right)} + \min\left(r, \frac{\log(1/\delta)}{\log n}\right)\frac{1}{1+\log(n/r)}\right)$$

Proof. Let t be an integer satisfying $\lceil \frac{n}{t} \rceil^{\lfloor \frac{r}{t} \rfloor} < \frac{1}{\delta}$; setting $t = \left\lceil \frac{r \log n}{\log \frac{1}{2\delta}} \right\rceil$ suffices, because

$$\log\left(\left\lceil \frac{n}{t} \right\rceil^{\lfloor \frac{r}{t} \rfloor}\right) \leq \left\lfloor \frac{r}{t} \right\rfloor \log \left\lceil \frac{n}{t} \right\rceil \leq \frac{r}{t} \log n \leq \frac{r}{\lceil r \log n / \log(1/2\delta) \rceil} \log n \leq \frac{\log(1/2\delta)}{\log n} \log n < \log \frac{1}{\delta} \,.$$

Note also that because $\delta \leq 1/(2n)$, $t \leq r$, and $\lfloor \frac{r}{t} \rfloor \geq 1$.

Given a randomized algorithm Π that solves $\textsc{mif}(n, r)$ with error $\leq \delta$ on any input stream, we will show how to construct a randomized algorithm Ψ which solves $\textsc{mif}(\lceil n/t \rceil, \lfloor r/t \rfloor)$ with the same error probability. As there are only $\lceil n/t \rceil^{\lfloor r/t \rfloor}$ possible input streams for the $\textsc{mif}(\lceil n/t \rceil, \lfloor r/t \rfloor)$ task, the probability (over randomness used by Ψ) of the event E than an instance A of Ψ succeeds on *any* of the streams in $[\lceil n/t \rceil]^{\lfloor r/t \rfloor}$ is $\geq 1 - \delta \lceil \frac{n}{t} \rceil^{\lfloor \frac{r}{t} \rfloor} > 0$. Therefore, by fixing the random bits of Ψ to some value for which the event E occurs, we obtain a deterministic protocol Φ for $\textsc{mif}(\lceil n/t \rceil, \lfloor r/t \rfloor)$.

We now explain the construction of Ψ given Π. Let $f : [n] \mapsto [\lceil n/t \rceil]$ be the function given by $f(x) = \lfloor x/t \rfloor$. For any $y \in [\lceil n/t \rceil]$, we have that $f^{-1}(y)$ is a nonempty set of size $\leq t$. The protocol Ψ starts by initializing an instance A of Π, and sending it $r - t\lfloor r/t \rfloor$ arbitrary stream elements.

When Ψ receives an element $e \in [\lceil n/t \rceil]$, it sends a sequence of t elements of $[n]$ to A, namely, the elements of $f^{-1}(e)$, in arbitrary order, repeating elements if $|f^{-1}(e)| < t$. To output an element, Ψ queries A to obtain $i \in [n]$, and reports $f(i)$. Assuming A did not fail, $f(i)$ is guaranteed to be a correct answer. If we assume for sake of contradiction that $f(i) = e$ for some element e sent to Ψ earlier, then A must have been sent all elements in $f^{-1}(e)$ – which implies that $i \in f^{-1}(e)$ and that A gave an incorrect output, contradicting the assumption that $f(i) = e$. Thus, we have proven that Ψ fails with no greater probability than Π, which is all that is needed to complete this part of the proof.

Having shown that $S_\delta(\textsc{mif}(n, r)) \geq S^{\text{DET}}(\textsc{mif}(\lceil \frac{n}{t} \rceil, \lfloor \frac{r}{t} \rfloor))$, we now substitute in the lower bound from Theorem 7.1.

$$S_\delta(\textsc{mif}(n, r)) \geq S^{\text{DET}}\left(\textsc{mif}\left(\left\lceil \frac{n}{t} \right\rceil, \left\lfloor \frac{r}{t} \right\rfloor\right)\right) \geq \Omega\left(\max\left(\sqrt{\left\lfloor \frac{r}{t} \right\rfloor}, \frac{\lfloor r/t \rfloor}{1 + \log(\lceil n/t \rceil / \lfloor r/t \rfloor)}\right)\right)$$

Because $\lfloor r/t \rfloor = \Theta\left(\min\left(r, \frac{\log(1/\delta)}{\log n}\right)\right)$, and $\lceil n/t \rceil / \lfloor r/t \rfloor = \Theta(n/r)$, and $\Omega(\max(a, b)) = \Omega(a + b)$, this simplifies to:

$$S_\delta(\textsc{mif}(n, r)) = \Omega\left(\sqrt{\min\left(r, \frac{\log(1/\delta)}{\log n}\right)} + \min\left(r, \frac{\log(1/\delta)}{\log n}\right) \frac{1}{1 + \log(n/r)}\right)$$

□

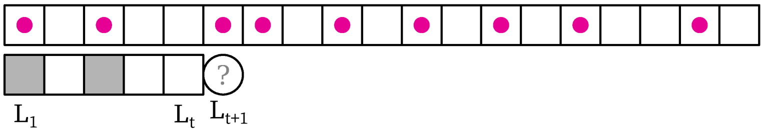

Figure 1: This diagram shows the behavior of Algorithm 4.1 on an example input. The top row of squares corresponds to the set $[n]$, ordered so that the leftmost squares corresponds to the elements $L_1, L_2, \ldots, L_{t+1}$ from Algorithm 4.1. In the top row, cells contain a pink dot if the corresponding element has already been seen in the stream. In the bottom row, each of the cells is shaded dark if the corresponding entry in the vector x is equal to 1 – except for L_{t+1}, whose state Algorithm 4.1 does not track.

4.1 Upper bound: a sampling algorithm

Theorem 4.2. *Algorithm 4.1 solves* $\textsc{mif}(n, r)$ *with error* $\leq \delta$ *on any fixed input stream, and uses* $s \leq \min(r, \frac{\log(1/\delta)}{\log(n/r)})$ *bits of space. (This assumes oracle access to* $O((s + 1) \log n)$ *random bits.)*

Algorithm 4.1. A streaming algorithm for $\textsc{mif}(n, r)$ with error rate $\leq \delta$ on any input stream

```
    Let $t = \min(r, \lfloor \log(1/\delta)/\log(n/r) \rfloor)$

    Initialization:
 1: Let $L = \{L_1, \ldots, L_{t+1}\}$ be a fixed sequence of elements in $[n]^{t+1}$ without repetitions, chosen uniformly at
    random. (This can be stored explicitly using $O((t+1)\log n)$ bits, or computed on demand as a function of
    $O((t+1)\log n)$ oracle random bits.)
 2: $x \leftarrow \{0, \ldots, 0\}$, a vector in $\{0,1\}^t$

    Update($e \in [n]$):
 3: if $\exists j \in [t] : L_j = e$ then
 4:     $x_j \leftarrow 1$
 5: end if

    Query:
 6: if $\exists j \in [t] : x_j = 0$ then
 7:     output: $L_j$
 8: else
 9:     output: $L_{t+1}$
10: end if
```

Proof. First, we observe that Algorithm 4.1 gives an incorrect output only when the input stream $\sigma = (e_1, \ldots, e_r)$ contains every element of L. Otherwise, either the first t elements of L are in σ, and L_{t+1} isn't – in which case Line 9 returns L_{t+1} – or there is some $j \in [t]$ where L_j has not been seen in the stream so far, in which case Line 7 correctly returns L_j. Given a fixed input stream $\sigma \in [n]^r$, the probability that Algorithm 4.1 fails is:

$$\Pr[L \subseteq \sigma] = \binom{|\sigma|}{t+1} \Big/ \binom{n}{t+1} \le \binom{r}{t+1} \Big/ \binom{n}{t+1} = \frac{r(r-1)\cdots(r-t)}{n(n-1)\cdots(n-t)} \le \left(\frac{r}{n}\right)^{t+1}$$

Thus $\Pr[L \subseteq \sigma]$ is $\le \delta$ when $t = \lfloor \log(1/\delta)/\log(n/r) \rfloor$, and is equal to 0 when $t = r$, because no set of size r can contain a set of size $r+1$. ∎

5 Adversarially robust model

THEOREM 5.1. *Any algorithm which solves* MIF(n, r) *against adaptive adversaries with total error δ requires $\ge \log(\binom{n}{\lceil r/2 \rceil}/\binom{n-\lceil r/2 \rceil}{\lfloor r/2 \rfloor + 1})) + \log(1-\delta)$ bits of space; or less precisely, $\Omega(r^2/n + \log(1-\delta))$.*

Proof. We prove this by reducing the communication task AVOID$(n, \lceil r/2 \rceil, \lfloor r/2 \rfloor + 1)$ (see Section 3) to MIF(n, r).

Say Alice is given the set $A \subseteq [n]$ of size $\lceil r/2 \rceil$. They instantiate an instance $\mathcal{X}$ of the given algorithm for MIF(n, r), and runs it on the partial stream of length $\lceil r/2 \rceil$ containing the elements of A in some arbitrary order. Alice then sends the state of $\mathcal{X}$ to Bob; since this is a public coin protocol, all randomness can be shared for free. Bob then runs the following adversary against $\mathcal{X}$; it queries $\mathcal{X}$ for an element b_0, and then provides that element back to $\mathcal{X}$, repeating this process $\lfloor r/2 \rfloor + 1$ times to recover elements $b_0, b_1, \ldots, b_{\lfloor r/2 \rfloor}$. The instance will fail to give correct answers to this adversary with total probability $\le \delta$. If it succeeds, then by the definition of the Missing Item Finding problem, $b_0 \notin A$, $b_1 \notin \{b_1\} \cup A$, and so on; thus Bob can report $B := \{b_0, \ldots, b_{\lfloor r/2 \rfloor + 1}\}$ as a set of $\lfloor r/2 \rfloor + 1$ elements which are disjoint from A.

This AVOID protocol implementation uses the same number of bits of communication as $\mathcal{X}$ does of space. By Lemma 3.1, it follows $\mathcal{X}$ needs:

$$\begin{aligned} &\ge \log\left(\binom{n}{\lceil r/2 \rceil} \Big/ \binom{n - \lfloor r/2 \rfloor - 1}{\lceil r/2 \rceil} \right) + \log(1-\delta) \\ &\ge \frac{\lceil r/2 \rceil (\lfloor r/2 \rfloor + 1)}{n \ln 2} + \log(1-\delta) \ge \frac{r^2}{4n \ln 2} + \log(1-\delta)\,, \end{aligned}$$

bits of space. ∎

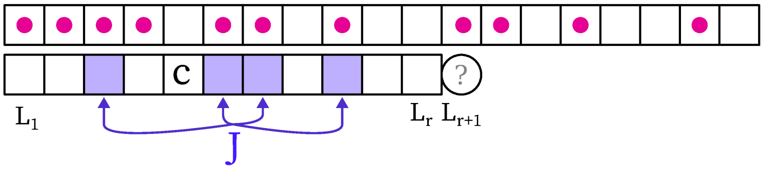

Figure 2: This diagram shows the behavior of Algorithm 5.1 on an example input. The top row of squares corresponds to the set $[n]$, ordered so that the leftmost squares corresponds to the elements L_1, L_2, …, L_{r+1} from Algorithm 5.1. In the top row, cells contain a pink dot if the corresponding element has already been seen in the stream. In the bottom row, the letter C indicates the cell corresponding to L_c. Cells that are shaded dark blue indicate the values contained in J. The third cell from the left is included in J because, at the time the element L_3 was added by the adversary, c was less than or equal to 2.

5.1 Upper bound: the hidden list algorithm

Theorem 5.2. *Algorithm 5.1 solves* MIF(n, r) *against adaptive adversaries, with error* δ*, and can be implemented using* $O(\min(r, \left(1 + \frac{r^2}{n} + \ln\frac{1}{\delta}\right) \cdot \log r))$ *bits of space. (It assumes oracle access to* $(r+1)\log n$ *random bits.)*

Algorithm 5.1. An adversarially robust algorithm for MIF(n, r) with error $\le \delta$

```
Let $t = \min(r, \lceil 3\frac{r^2}{n} + \ln\frac{1}{\delta} \rceil)$

Initialization:
Let $L = \{L_1, \ldots, L_{r+1}\}$ be a fixed sequence of elements in $[n]^{r+1}$ without repetitions, chosen uniformly at random. (Assuming oracle access to $O(r \log n)$ random bits, the value of $L$ can be computed on demand, instead of stored.)
$c \leftarrow 1$, an integer in the range $\{1, \ldots, r+1\}$
$J \leftarrow \emptyset$, a subset of $\{L_1, \ldots, L_r\}$ of size $\le t$

Update($e \in [n]$):
while $e = L_c$ or $L_c \in J$ do
    $c \leftarrow c + 1$
end while
if $e \in \{L_{c+1}, \ldots, L_r\}$ then
    $J \leftarrow J \cup \{e\}$
end if
if $|J| > t$ then
    abort
end if

Query:
output: $L_c$
```

Proof. First, we observe that the only way that Algorithm 5.1 can fail is if it aborts. At any point in the stream, the set J includes the intersection of the earlier elements from the stream, with the list $\{L_{c+1}, \ldots, L_r\}$ of possible future outputs. The while loop ensures that the element L_c emitted will neither be equal to the current element nor collide with any past stream elements (those in J). It is not possible for c to go out of bounds, because each element in the stream can lead to an increase in c of at most one; either immediately when the element arrives, if $e = L_c$; or delayed slightly, if $e \in \{L_{c+1}, \ldots, L_r\}$. Since the stream contains r elements, c will increase by at most r, to a value of $r + 1$. Note that if c has reached the value $r + 1$, then the entire stream was a permutation of $\{L_1, \ldots, L_r\}$, making L_{r+1} is a safe output.

This algorithm needs $\log(r + 1)$ bits to store c, but the main space usage is in storing J. We will show that $|J| \le t$ with probability $\ge 1 - \delta$, in which case J can be stored as either a bit vector of length r, or a list of $\le t$ indices in $[r]$, using $O(\min(r, t \log r))$ bits of space.

We observe that after $i-1$ elements have been received (and up to i distinct elements emitted), the probability that the ith element chosen by the adversary will be newly stored in J will be $\le 2\frac{r}{n}$, no matter what the earlier elements were or what the adversary picks. If $r \ge n/2$, this is immediate. Otherwise, write E_{i-1} for the set containing the first $i-1$ elements of the stream, e_i for the ith element, and let c_i be the value of the variable c as of Line 7. Let X_i denote the indicator random variable for the event that e_i was not in J before, but has been added now.

Because the adversary has only been given outputs deriving from $L_{\le c_i} := (L_1, \dots, L_{c_i})$, if we condition on the random variable $L_{\le c_i}$, then the element e_i and set E_{i-1} are independent of $L_{>c_i} := \{L_{c_i+1}, \dots, L_r\}$. Given E_{i-1}, the values $X_1, \dots, X_{i-1}$ determine whether or not each element of E_{i-1} is in $L_{>c_i}$. Then, conditioning on $L_{\le c_i}, e_i, E_{i-1}$, and $X_1, \dots, X_{i-1}$, we have that $L_{>c_i} \setminus E_{i-1}$ is a set of size $r - c_i - |L_{>c_i} \cap E_{i-1}|$ chosen uniformly at random from $[n] \setminus L_{\le c_i} \setminus E_{i-1}$. Thus, if $e_i \notin L_{\le c_i} \cup E_{i-1}$, the probability that $X_i = 1$ is precisely the probability that e_i is contained in $L_{>c_i} \setminus E_{i-1}$, so:

$$\begin{aligned}\Pr\left[X_i = 1 \mid (X_j)_{j=1}^{i-1}, e_i, E_{i-1}, L_{\le c_i}, \{e_i \notin L_{\le c_i} \cup E_{i-1}\}\right] &= \frac{r - c_i - |L_{>c_i} \cap E_{i-1}|}{n - c_i - |E_{i-1} \setminus L_{\le c_i}|} \\ &\le \frac{r - c_i}{n - c_i - r} \le \frac{r}{n-r} \le \frac{2r}{n} \,.\end{aligned}$$

On the other hand, the event $e_i \in L_{\le c_i} \cup E_{i-1}$, implies $X_i = 0$ always. Together, these imply $\Pr[X_i = 1 \mid (X_j)_{j=1}^{i-1}] \le 2r/n$.

Then applying the (modified, see Lemma 3.3) Azuma's inequality bound, we find that with $z := \max\{1, \frac{3n}{2r^2} \ln \frac{1}{\delta}\}$:

$$\begin{aligned}\Pr[\sum_{i \in [r]} X_i \le \frac{2r^2}{n}(1+z)] &\le e^{-\frac{z}{2+z} z \frac{2r^2}{n}} \\ &\le e^{-z \frac{2r^2}{3n}} && \text{since } z \ge 1 \\ &\le e^{-\ln\frac{1}{\delta}} = \delta \,. && \text{since } z \ge \frac{3n}{2r^2} \ln \frac{1}{\delta}\end{aligned}$$

This implies that the probability that $|J|$ exceeds $2r^2/n + 3\ln(1/\delta)$ will be $\le \delta$. Consequently, our bound for the total space usage of the algorithm is:

$$\begin{aligned}&O(\log r) + O(\min(r, \left(\frac{r^2}{n} + \ln\frac{1}{\delta}\right) \log r)) \\ &\qquad = O(\min(r, \left(1 + \frac{r^2}{n} + \ln\frac{1}{\delta}\right) \cdot \log r))\end{aligned}$$

□

While it is possible to reduce the space usage of Algorithm 5.1 by removing all elements from the set J that are less or equal than c, this only changes the constant factor.

6 Zero error model

Theorem 6.1. *All algorithms solving* MIF(n, r) *with zero error on any stream require* $\Omega(r^2/n)$ *bits of space, in expectation over the randomness of the algorithm.*

Proof. First, we prove that if there is a zero-error algorithm Φ for MIF(n, r) using exactly s bits, in expectation, then there is a communication protocol for AVOID$(n, \lceil r/2 \rceil, \lfloor r/2 \rfloor + 1)$ using prefix-encoded messages with an expected length of s bits. The construction is the same as for Theorem 5.1. Alice, on being given a set $A \subseteq [n]$ of size $\lceil r/2 \rceil$, initializes an instance X of Φ, and runs it on an input stream α of length $\lceil r/2 \rceil$ containing each element of A in some arbitrary order. Any random bits used by X are shared publicly with Bob. They send the encoding of X's state to Bob, who queries X to find an element $b_0 \notin \alpha$, updates X with b_0, queries it to find $b_1 \notin \alpha \cup \{b_0\}$, and so on until Bob has recovered $B = \{b_0, \dots, b_{\lfloor r/2 \rfloor}\}$. Because the algorithm is guaranteed to never fail on any input stream, it must in particular succeed on Bob's adaptively chosen continuation of α. This ensures that $B \cap A = \emptyset$ holds with probability 1.

Next, we prove that any zero error randomized communication protocol Π for $\textsc{avoid}(t, a, b)$ requires $\geq ab/(t \ln 2)$ bits *in expectation*. Following the argument from Lemma 6 of [6], we observe that there must exist a fixing of the public randomness of Π for which the expected number of bits used when inputs A are drawn uniformly at random from $\binom{[t]}{a}$, is at least as large as when Π is run unmodified. Let Υ be the deterministic protocol with this property, and let M be the set of all messages sent by Υ. Each message $m \in M$ has a length $|m|$, probability (over the random choice of A) p_m of being sent, and makes Bob output the set B_m. For all $m \in M$, we have:

$$p_m = \Pr[m \text{ is sent}] \leq \Pr[B_m \text{ is a correct output}] = \Pr[A \cap B_m = \emptyset] \leq \binom{t-a}{b} / \binom{t}{a} \leq 2^{-\frac{ab}{t \ln 2}}.$$

Let $\Upsilon(A) \in M$ be the message sent by Υ for a given value of A. Then the entropy

$$H(\Upsilon(A)) = \sum_{m \in M} p_m \log \frac{1}{p_m} = \mathbb{E}_{A \in \binom{[t]}{a}} \log \frac{1}{p_{\Upsilon(A)}} \geq \mathbb{E}_{A \in \binom{[t]}{a}} \frac{ab}{t \ln 2} = \frac{ab}{t \ln 2}\,.$$

By the source coding theorem,

$$\mathbb{E}_{A \in \binom{[t]}{a}} \mathbb{E} |\Upsilon(A)| \geq H(\Upsilon(A)) \geq \frac{ab}{t \ln 2}\,.$$

Applying the above lower bound to the task $\textsc{avoid}(n, \lceil r/2 \rceil, \lfloor r/2 + 1 \rfloor)$, we conclude that Φ requires $\geq \frac{r^2}{4n \ln 2}$ bits of space in expectation.

□

Theorem 6.2. *There is an algorithm solving* $\textsc{mif}(n, r)$ *with zero error against adaptive adversaries, which uses* $O((1 + r^2/n) \log r)$ *bits of space, in expectation over the randomness of the algorithm.*

Proof. We use a slight variation of Algorithm 5.1, in which internal parameter t is instead set to r. This ensures that the algorithm will never abort; the proof of Theorem 5.2 has established that Algorithm 5.1 will then always give a correct output for the $\textsc{mif}(n, r)$ task.

The counter c can be encoded in binary using at most $\lceil \log(r+1) \rceil$ bits. We encode the set J by concatenating the binary value of $|J|$, followed by the binary values of the indices $i_1, \ldots, i_{|J|}$ in $[r]$ for which L_{i_k} is equal to the kth smallest element of J. (As both the encodings of c and J are prefix codes, so too is the encoding of the algorithm's state formed by concatenating them.) The total space S used by the algorithm (excluding random bits) is then:

$$S = \lceil \log(r+1) \rceil + \lceil \log r \rceil (1 + |J|)\,.$$

As in the proof of Theorem 5.2, let X_i be the indicator random variable for the event that the ith element that the adversary chooses for the stream is stored in J; we showed that for all $i \in [r]$, $\Pr[X_i = 1 \mid X_{i-1}, \ldots, X_1] \leq \frac{r-1}{n}$, which implies $\mathbb{E}[X_i] \leq \frac{r-1}{n}$. By linearity of expectation,

$$\begin{aligned} \mathbb{E} S &= \lceil \log(r+1) \rceil + \lceil \log r \rceil \left(1 + \mathbb{E} \sum_{i \in [r]} X_i \right) \\ &\leq \lceil \log(r+1) \rceil + \lceil \log r \rceil \left(1 + \frac{r(r-1)}{n} \right) = O((1 + \frac{r^2}{n}) \log r)\,. \end{aligned}$$

□

7 Deterministic model

7.1 Lower bound: an embedded instance of Avoid

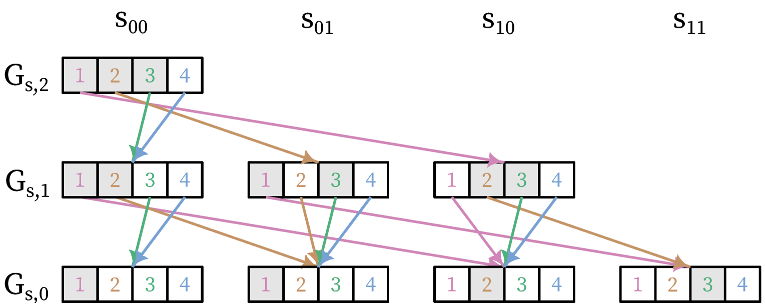

Figure 3: In the proof of Theorem 7.1, the quantities F_σ (defined in Eq. 7.1) are entirely determined by the values of $\Sigma[\sigma]$ and $r - |\sigma|$. More precisely, we have $F_\sigma = G_{\Sigma[\sigma], r-|\sigma|}$, where $G_{s,i} := \{\omega_x : \exists \alpha \in [n]^i : \tau(s, \alpha) = x\}$. This diagram shows the values of $G_{s,i}$ for Algorithm 3.1 solving the MIF$(4, 2)$ problem. The sets $G_{s,i}$ are represented by the dark squares in the array of four cells. The transition function between states is indicated by the colored arrows; for example, green colored arrows (those emitting from squared numbered with a 3) correspond to transitions where the next stream element is a 3, i.e, from state s to state $s' = \tau(s, 3)$.

Theorem 7.1. *Every deterministic streaming algorithm for* MIF(n, r) *requires* $\Omega(\sqrt{r} + \frac{r}{1+\log(n/r)})$ *bits of space.*

Proof. Let Σ be the set of states of the algorithm, and let s_{init} be the initial state. Let $\tau : \Sigma \times [n]^\star \mapsto \Sigma$ be the transition function of the algorithm, where $\tau(s, e_1, \dots, e_k) = x$ means that if the algorithm is at state s, and the next k elements in the stream are $e_1, \dots, e_k$, then after processing those elements the algorithm will reach state x. For each partial stream $\sigma \in [n]^\star$, abbreviate $\tau(s_{\text{init}}, \sigma)$ as $\Sigma[\sigma]$. For each state $s \in \Sigma$, we associate the output $\omega_s \in [n]$ which the algorithm would emit if the state is reached at the end of the stream. (If there is no stream of length r leading to state s, we let ω_s be arbitrary.)

For each partial stream $\sigma \in [n]^\star$, let

$$F_\sigma = \{i : \exists x \in \Sigma, \exists \alpha \in [n]^{r-|\sigma|} : \tau(\Sigma[\sigma], \alpha) = x \wedge \omega_x = i\} \tag{7.1}$$

be the set of possible outputs of the algorithm when σ is extended to a stream of length r. Because there are only $|\Sigma|$ states, and only $[n]$ possible output values, $|F_s^i| \le m$, where $m = \min(|\Sigma|, n)$.

Let t, q be integers chosen later, so that

$$tq \le r - m/2^q \,. \tag{7.2}$$

We claim that there exists a partial stream $\sigma \in [n]^\star$ satisfying $\forall \alpha \in [n]^t : |F_{\sigma.\alpha}| \ge \frac{1}{2}|F_\sigma|$.

Such a state can be found by an iterative process. Let τ_0 be the empty stream ϵ; for $i = 1, 2, 3, \dots.$, if there exists $\alpha \in [n]^t$ for which $|F_{\tau_i.\alpha}| \le \frac{1}{2}|F_{\tau_i}|$, let $\tau_{i+1} = \tau_i.\alpha$. Otherwise, stop, and let $\sigma = \tau_i$. This process must terminate before $i = q$, because otherwise we would have $|F_{\tau_q}| \le m/2^q \le r - qt$. Then letting $\gamma \in [n]^{r-qt}$ be a sequence of elements containing every element of F_{τ_q}, we observe that the algorithm cannot possibly output a correct answer for the stream $\tau_q.\gamma$. By the definition of F_{τ_q}, we must have $\omega_{\tau_q.\gamma} \in F_{\tau_q}$; but to be a correct missing item finding solution, we need $\omega_{\tau_q.\gamma} \notin \gamma$, hence $\omega_{\tau_q.\gamma} \notin F_{\tau_q}$, a contradiction. Thus, $\sigma = \tau_i$ for some $i \le q - 1$. Thus $|\sigma| \le (q-1)t \le r - t$, which ensures that the terms $\sigma.\alpha$ are streams of length $\le r$ and therefore well defined. Finally, the stopping condition of the process implies $\forall \alpha \in [n]^t : |F_{\sigma.\alpha}| \ge \frac{1}{2}|F_\sigma|$.

We will now construct a deterministic protocol for AVOID$(|F_\sigma|, t, \lceil \frac{1}{2}|F_\sigma| \rceil)$ using $\le \log|\Sigma|$ bits of communication. Alice, on being given a set $A \in \binom{F_\sigma}{t}$, arbitrarily orders it to form a sequence α in $(F_\sigma)^t$; and then sends the state $s' = \tau(\Sigma[\sigma], \alpha)$ to Bob. This can be done using $\log|\Sigma|$ bits of space. Bob uses the encoded state to find $F_{\sigma.\alpha}$, by evaluating $\omega_{\tau(s',\beta)}$ for all sequences $\beta \in [n]^{|\sigma|-t}$, and reports the first $\lceil \frac{1}{2}|F_\sigma| \rceil$ elements of this set as B. This protocol works because as claimed above, we are guaranteed $|F_{\sigma.\alpha}| \ge |F_\sigma|$; and furthermore, $F_{\sigma.\alpha}$ must be disjoint from A; if it is not, then there exists some continuation of σ concatenated with α which leads the algorithm to a state z with $\omega_z \in A$, contradicting the correctness of the MIF protocol. Finally, applying the communication lower bound from Lemma 3.1, we find

$$\log|\Sigma| \ge \frac{1}{\ln 2} t \left\lceil \frac{1}{2}|F_\sigma| \right\rceil / |F_\sigma| \ge t/(2\ln 2) \tag{7.3}$$

We now determine values of t and q satisfying Eq. 7.2. We can set

$$q = \lceil 1 + \log(m/r) \rceil \qquad \text{and} \qquad t = \left\lfloor \frac{1}{q}\left(r - \frac{m}{2^q}\right) \right\rfloor$$

We must have $m \geq r+1$, as otherwise $|F_\epsilon| \leq m \leq r$, in which case we could easily make the algorithm give an incorrect output by running it on a stream $\gamma \in [n]^r$ which contains all elements of F_ϵ. Thus $\log(m/r) \geq 0$, and hence $q \geq 1$, making t well defined. Since $m = \min(|\Sigma|, n)$, we are also guaranteed $\log|\Sigma| \geq \log(r+1)$. Combining this with Eq. 7.3 gives:

$$\begin{aligned}
\log|\Sigma| &\geq \max\left(\log(r+1), \frac{1}{2\ln 2}\left\lfloor \frac{1}{q}\left(r - \frac{m}{2^q}\right)\right\rfloor\right) \\
&\geq \max\left(1, \frac{1}{2\ln 2}\left\lfloor \frac{r}{2q}\right\rfloor\right) && \text{since } 2^q \geq 2m/r \text{ and } r \geq 1 \\
&\geq \frac{1}{1+2\ln 2}\cdot\frac{r}{2q} && \text{since } \min(1,(z-1)/y) \geq \frac{z}{1+y} \\
&\geq \frac{r}{10+5\log(m/r)}\,. && \text{since } 1+2\ln 2 \leq 5/2
\end{aligned} \tag{7.4}$$

As $m = \min(|\Sigma|, n)$, we have $m \leq |\Sigma|$, so

$$\log|\Sigma| \geq \frac{r/5}{2+\log|\Sigma| - \log r} \qquad \Longrightarrow \qquad (\log|\Sigma|)^2 + (2-\log r)\log|\Sigma| - r/5 \geq 0\,.$$

Solving the quadratic inequality gives:

$$\log|\Sigma| \geq \sqrt{\frac{r}{5} + \left(1 - \frac{\log(r)}{2}\right)^2} - \left(1 - \frac{\log(r)}{2}\right) \geq \begin{cases} \sqrt{r/5} & \text{if } r \geq 4 \\ 0 & \text{otherwise} \end{cases}$$

As $\log|\Sigma| \geq \log(r+1) \geq \sqrt{r/5}$ also holds for $r \leq 4$, it follows that $\log|\Sigma| \geq \sqrt{r/5}$ for all values of r. Combining this result, Eq. 7.4, and the inequality $m \leq n$, we conclude:

$$\log|\Sigma| \geq \max\left(\sqrt{\frac{r}{5}}, \frac{r}{10+5\log(n/r)}\right) = \Omega\left(\sqrt{r} + \frac{r}{1+\log(n/r)}\right)\,.$$

□

Note: instead of associating "forward" looking sets of outputs F_s with each state $s \in \Sigma$, we could instead use "backward" looking states B_s defined (roughly) as $[n] \setminus \{i : \exists\sigma \text{ leading to } s \text{ with } i \in \sigma\}$.

7.2 Upper bound: the iterated pigeonhole algorithm

Theorem 7.2. *Algorithm 7.1 is a deterministic algorithm that solves* MIF(n,r) *using* $O(\sqrt{r\log r} + \frac{r\log r}{\log n})$ *bits of space.*

Algorithm 7.1. A deterministic algorithm for MIF(n,r)

```
Let $s, t$ be integers satisfying $s^t \leq n$, and $t(s-1) \geq r$.

Initialization:
$x \leftarrow (0,\ldots,0)$ is a vector in $\{0,1\}^s$.
$(\ell, a) \leftarrow (1,0)$ is an element of $\bigcup_{j\in[t]}\{j\}\times\{0,\ldots,s^j-1\}$

Update($e \in [n]$):
Let $i \leftarrow \left(\lfloor (e-1)/s^{\ell-1}\rfloor \bmod s\right) + 1$
$x_i \leftarrow 1$
```

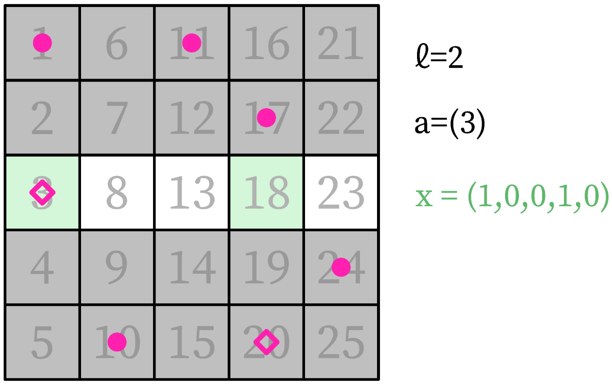

Figure 4: This diagram shows the behavior of Algorithm 7.1, with $s = 5$ and $t = 2$, on an example input. The pink circles and diamonds mark the elements currently covered by the stream. Cells shaded dark gray are those which are no longer possible outputs due to the current values of ℓ and a. Cells shaded light green are no longer possible outputs due to the value of the vector x. Cells shaded white are possible output values. The algorithm proceeds in t phases; in this example, for the first phase, it maintained a bit vector tracking which of the s rows of the set $[n]$ contained an element from the stream; after the first five elements (1, 10, 11, 17, 24 in some order) arrived, only one row was left available, and the algorithm proceeded to the second phase – maintaining a bit vector x that records which columns within the chosen row may be unavailable.

```
 5: if $\ell < t$ and there is exactly one $y \in [s] : x_y = 0$ then
 6:     $x \leftarrow (0, \dots, 0)$
 7:     $\ell \leftarrow \ell + 1$
 8:     $a \leftarrow a + (y - 1)s^{\ell-1}$
 9: end if

    Query:
10: Let $i$ be the least value in $[s]$ for which $x_i = 0$
11: output: $a + (i - 1)s^{\ell-1} + 1$
```

Proof. First, we establish that the variables (ℓ, a) of the algorithm stay in their specified bounds. The condition in Line 5 ensures that ℓ will not be increased beyond t. At the time Line 8 is executed, $a < s^{\ell-1}$; since $y \in [s]$, it follows $a + (y - 1)s^{\ell-1} < (1 + (s - 1))s^{\ell-1} \leq s^\ell$, so the pair (ℓ, a) stays in $\bigcup_{j\in[t]}\{j\} \times \{0, \dots, s^j - 1\}$.

Next, we establish that the algorithm is correct; that the output from Line 11 does not overlap with current stream $e_1, \dots, e_k$. For each element e_j in the stream, let ℓ_j be the value of ℓ at the time the element was added (i.e., at the start of the Update function). For all $h \in [t]$, define $C_h := \{j \in [t] : \ell_j = h\}$ to indicate the elements for which $\ell_j = h$. Because Line 5 only triggers when x has one zero entry, and x is reset to the all zero vector immediately afterwards, and each new element sets at most one entry of x to 1 (Line 4), we have $|C_h| \geq s - 1$ for all h less than or equal to the current value of ℓ.

Let $c = a+(i-1)s^{\ell-1}$ be the current output of the algorithm (Line 11), minus 1. Note that $c \leq s^t-1 \leq n-1$, so the output is in $[n]$. The value of c can be written in base s as $(c_1, \dots, c_t)$, so that $c = \sum_{j=1}^{t} c_j s^{j-1}$. For h less than the current value of ℓ, c_h is equal to the value of y at the time the condition of Line 5 evaluated to true; in other words, at that time, $x_{c_h} = 0$. Now, for each $j \in C_h$, consider the value $e_j - 1$ written in base s as $(b_1, \dots, b_t)$. When e_j was added, Line 3 set i equal to b_h, and so Line 4 ensured $x_{b_h} = 1$. Since $x_{c_h} = 0$ held afterwards, when the condition of Line 5 evaluated to true, it follows $b_h \neq c_h$. This implies $e_j - 1 \neq c$ holds for all $j \in C_h$. A similar argument will establish that for $j \in C_\ell$, we have $e_j - 1 \neq c$; since $C_1 \cup \dots \cup C_\ell$ contains the entire stream so far, it follows the current output of the algorithm does not equal any of the $\{e_j\}_{j=1}^{k}$, and is thus correct.

Finally, we determine values of s and t which for which the algorithm uses little space. The vector x can be stored using s bits; since there are $\sum_{i=0}^{t-1} s^i \leq s^t$ possible values of (ℓ, a), this algorithm can be implemented using $\leq s + t \log s + 1$ bits of space.

Now let

$$q = \min\left(\sqrt{r\log(r+1)}, \log n\right) \qquad \text{and} \qquad t = \left\lfloor \frac{q}{\log(r+1)} \right\rfloor \qquad \text{and} \qquad s = \left\lceil \frac{r}{t} \right\rceil + 1\,,$$

Because $r \geq \log(r+1)$, and $\log n \geq \log(r+1)$, it follows $t \geq 1$. This implies $s \leq r+1$. Then $t(s-1) = t\lceil r/t \rceil \geq r$, and

$$s^t \leq (r+1)^{\lfloor q/\log(r+1)\rfloor} \leq (r+1)^{q/\log(r+1)} \leq 2^q \leq n\,,$$

so the values of s and t satisfy the required conditions $s^t \leq n$ and $t(s-1) \geq r$. With these parameters, the space usage of the algorithm is:

$$\begin{aligned} s + t\log s + 1 &\leq \left\lceil \frac{r}{t} \right\rceil + 2 + \left\lfloor \frac{q}{\log(r+1)} \right\rfloor \log(\left\lceil \frac{r}{t} \right\rceil + 1) \\ &\leq \frac{r}{\lfloor q/\log(r+1)\rfloor} + 3 + \frac{q}{\log(r+1)} \log(r+1) \\ &\leq \frac{2r\log(r+1)}{q} + q + 3 \\ &= \max\left(2\sqrt{r\log(r+1)}, \frac{2r\log(r+1)}{\log n}\right) + \min\left(\sqrt{r\log(r+1)}, \log n\right) + 3 \\ &= O\left(\sqrt{r\log r} + \frac{r\log r}{\log n}\right). \end{aligned}$$

□

8 White box model

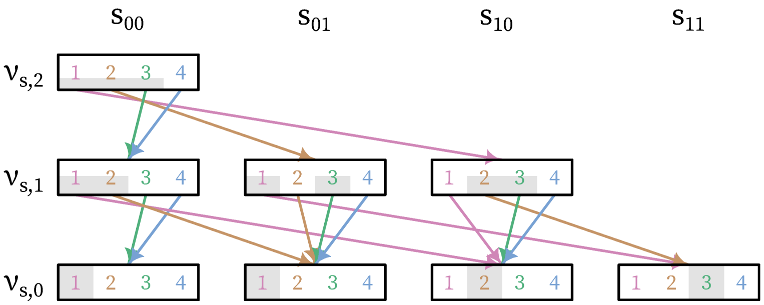

Figure 5: In the proof of Theorem 8.1, the quantities $\nu_{\sigma,i}$ defined as fixed points of Eq. 8.6 are shown for the state diagram of Algorithm 3.1 for the problem MIF$(4,2)$. The distributions $\nu_{\sigma,i}$ are represented by the gray bar charts in each rectangle; for example, the distribution $\nu_{s_{01},1}$ has weight 0.5 on value 1 and weight 0.5 on value 3. The transition function between states is indicated by the colored arrows; for example, green colored arrows (those emitting from squared numbered with a 3) correspond to transitions where the next stream element is a 3, i.e, from state s to state $s' = \tau(s,3)$.

THEOREM 8.1. *Every streaming algorithm for* MIF(n,r) *which has error* $\delta \leq 1/(16n)^{2\log n+7} = 1/2^{\Omega(\log n)^2}$ *against white-box adversaries requires* $\Omega\left(\frac{r}{(\log n)^4}\right)$ *bits of space.*

This proof relies on the following Lemma, whose proof we will defer for later.

LEMMA 8.1. *Let* ν *be a distribution on* $[n]$*, and* $p = 1 + 1/\log(n)$*. Let* $\delta \leq \frac{1}{n^3}$*. Let* R *be a random variable with values in* Ω*. If there is a map* $M : [n]^t \times \Omega \to \triangle[n]$ *so that* $\mathbb{E}_{x\sim\nu^t,R} M(x,R) = \nu$*, and:*

$$\Pr_{x\sim\nu^t,R}\left[\|M(x,R)\|_p^p \leq D^{p-1}\|\nu\|_p^p \wedge (\forall j \in [t] : M(x,R)(x_j) \leq \delta)\right] \geq 1 - \frac{1}{2^6 n^2} \tag{8.5}$$

Then $\log|\text{range}(M)| \geq \frac{t}{2^9 D(\log n)^2} - 2\log(n) - 6.$

Proof. Proof of Theorem 8.1.

We can safely assume that $r \geq \log n + 1$, as for any $r = O((\log n)^3)$ the claimed lower bound is trivial.

Let $\ell = \lceil \log n + 1 \rceil$, $t = \left\lfloor \frac{r}{\ell} \right\rfloor$, and let $\hat{r} = t\ell$. We can use a protocol for $\textsc{mif}(n, r)$ to solve $\textsc{mif}(n, \hat{r})$ instead, by padding the start of the stream with a fixed sequence of $r - \hat{r}$ arbitrary inputs. Let $\mathcal{A}$ be this new algorithm.

Let Σ be the set of all states of $\mathcal{A}$, and let $\tau : \Sigma \times [n] \to \Sigma$ be the *randomized* transition function between states. For each state $s \in \Sigma$, let ω_s be the distribution over $[n]$ from which the final output value is drawn when the final state of the algorithm is s. (If s can never occur at the end of the stream, we let ω_s be arbitrary.) To each pair $(s, i) \in \Sigma \times \{0, \ldots, \ell\}$, we will associate a distribution $\nu_{s,i}$ over $[n]$. These distributions are recursively defined; if $i = \ell$, we let $\nu_{s,\ell} = \omega_s$, i.e., the output distribution for state s. For $i < \ell$, define $f_{s,i} : \triangle[n] \to \triangle[n]$ as:

$$f(\phi) = \mathbb{E}_{x\sim\phi^t}\mathbb{E}_{s'\sim\tau(s,x)}\nu_{s',i+1} = \sum_{x\in[n]^t}\left(\prod_{i\in[t]}\phi(x_i)\right)\sum_{s'\in\Sigma}\Pr[\tau(s,x) = s']\nu_{s',i+1} \tag{8.6}$$

Because this function is continuous, and $\triangle[n]$ is homeomorphic to an $(n-1)$-dimensional ball, we can apply Brouwer's fixed point theorem (Lemma 3.4) to find a distribution $\nu_{s,i} \in \triangle[n]$ satisfying $\nu_{s,i} = f_{s,i}(\nu_{s,i})$.

With the distributions $\nu_{s,i}$ as defined above, we can define an adversary which, we can prove, will trick $\mathcal{A}$ into outputting an element that was present in the stream with probability $\geq \frac{1}{(16n)^{2\log n + 7}}$. The adversary proceeds in ℓ rounds: for each $i \in \{0, \ldots, \ell - 1\}$, they identify the current state s_i of the algorithm, sample $\alpha \sim \nu_{s_i,i}$, and send α to $\mathcal{A}$.

Let $p = 1 + 1/\log n$; the quantity $\|\nu_{s,i}\|_p^p$ is a measure of the concentration of the output distribution associated with s and i. Assume for sake of contradiction that $\log|\Sigma| \leq \frac{t}{2^9(\log n)^2} - 2\log n - 6$. Then we shall prove by induction, for all $i \in \{0, \ldots, \ell\}$, the statement $P(i)$ that for all $s \in \Sigma$, if $\|\nu_{s,i}\|_p^p \geq 2^{i(p-1)}/n^{p-1}$, then the probability that $\mathcal{A}$ will give an incorrect answer when the remaining $(\ell - i)t$ elements of the stream are provided by the adversary is $\geq 1/(16n)^{2(\ell-i)+3}$). The base case of the induction, at $i = \ell$, holds vacuously, because $\|\nu_{s,i}\|_p^p \geq 2^{i(p-1)}/n^{p-1} \geq 2^{(\lceil \log n \rceil + 1)(p-1)}/n^{p-1} \geq 2^{p-1} > 1$ is never true.

Now, for the induction step. Assume $P(i+1)$ holds; we would like to prove $P(i)$ is true. Assume the current state s of the algorithm satisfies $\|\nu_{s,i}\|_p^p \geq 2^{i(p-1)}/n^{p-1}$. The adversary samples $x \sim \nu_{s,i}^t$ and sends it to the algorithm, which transitions to the state $s' \sim \tau(s,x)$. If it is the case that

$$\Pr[\nu \text{ too concentrated}] := \Pr[\|\nu_{s',i+1}\|_p^p \geq 2^{(i+1)(p-1)}/n^{p-1}] \geq \frac{1}{2^7 n^2}\,, \tag{8.7}$$

then, by applying $P(i+1)$, it follows:

$$\begin{aligned}\Pr[\mathcal{A} \text{ fails}] &\geq \Pr[\mathcal{A} \text{ fails} \mid \nu \text{ too concentrated}]\Pr\left[\nu \text{ too concentrated}\right] \\ &\geq \frac{1}{(16n)^{2(\ell-i-1)+3}} \cdot \frac{1}{2^7 n^2} \geq \frac{1}{(16n)^{2(\ell-i)+3}}\end{aligned}$$

It remains to prove $P(i+1)$ assuming Eq. 8.7 does not hold. If that is the case, let $M : [n]^t \times \Omega \to \triangle[n]$ be the randomized map in which $M(x, R) = \nu_{s',i+1}$ where s' is randomly chosen according to $\tau(s,x)$; the random variable R encapsulates the randomness of τ. Note that $\mathbb{E}_{x\sim\nu_{s,i}^t,R}M(x,R) = \nu_{s,i}$, by the definition of $\nu_{s,i}$. Applying Lemma 8.1 to M, $\nu_{s,i}$, p, $D = 2$, and $\delta = 1/n^3$ we observe that since we have assumed that $|\Sigma| \geq \log|\operatorname{range}(M)|$ is smaller than the Lemma guarantees, and $\mathbb{E}_{x\sim\nu_{s,i}^t}\mathbb{E}M(x) = \nu_{s,i}$ holds, it must be that Eq. 8.5 is incorrect. Thus:

$$\Pr_{x\sim\nu^t}\left[\|M(x,R)\|_p^p \leq 2^{p-1}\|\nu_{s,i}\|_p^p \wedge (\forall j \in [t] : M(x,R)(x_j) \leq \frac{1}{n^3})\right] \leq 1 - \frac{1}{2^6 n^2}$$

and since Eq. 8.7 does not hold, we have

$$\Pr_{x\sim\nu^t}\left[\|\nu_{s',i+1}\|_p^p \leq 2^{p-1}\|\nu_{s,i}\|_p^p\right] \geq \Pr_{x\sim\nu^t}\left[\|\nu_{s',i+1}\|_p^p \leq 2^{(i+1)(p-1)}/n^{p-1}\right] \geq 1 - \frac{1}{2^7 n^2}$$

which implies:

$$\Pr_{x\sim\nu^t, s'\sim\tau(s,x)}\left[\exists j \in [t] : \nu_{s',i+1}(x_j) \geq \frac{1}{n^3}\right] \geq \frac{1}{2^7 n^2}\,.$$

The definition of $\nu_{s',i+1}$ ensures that $\nu_{s',i+1}$ is precisely the distribution of output values if the algorithm and adversary are run for $t(\ell - i - 1)$ steps starting from state s'. The probability that the algorithm fails because the final output overlaps with x is then

$$\begin{aligned}
\Pr_{x\sim\nu^t_{s,i},\, s'\sim\tau(s,x),\, y\sim\nu_{s',i+1}}[\exists j \in [t] : x_j = y] &= \mathbb{E}_{x\sim\nu^t_{s,i},\, s'\sim\tau(s,x)} \Pr_{y\sim\nu_{s',i+1}}[\exists j \in [t] : x_j = y] \\
&= \mathbb{E}_{x\sim\nu^t_{s,i},\, s'\sim\tau(s,x)} \sum_{j\in[t]} \nu_{s',i+1}(x_j) \\
&\geq \mathbb{E}_{x\sim\nu^t_{s,i},\, s'\sim\tau(s,x)} \max_{j\in[t]} \nu_{s',i+1}(x_j) \\
&\geq \frac{1}{n^3} \Pr_{x\sim\nu^t_{s,i},\, s'\sim\tau(s,x)} \left[\max_{j\in[t]} \nu_{s',i+1}(x_j) \geq \frac{1}{n^3}\right] \\
&\geq \frac{1}{n^3} \cdot \frac{1}{2^7 n^2} = \frac{1}{2^7 n^5}
\end{aligned}$$

Thus, the failure probability of the algorithm as of (s, i) is $\geq 1/(2^7 n^5) \geq 1/(16n)^5 \geq 1/(16n)^{2(\ell-i)+3}$; this completes the proof of $P(i)$.

With the proof by induction complete, the statement $P(0)$ implies that for any $s \in \Sigma$, because $\|\nu_{s,0}\|_p^p \geq \frac{1}{n^{p-1}}$ always holds, the probability that $\mathcal{A}$ gives an incorrect answer when run against the adversary on a stream of length $t\ell = \hat{r}$ is $\geq 1/(16n)^{2\ell+3} \geq \frac{1}{(16n)^{2\log n+7}}$. This contradicts the given fact that $\mathcal{A}$'s error is less than this, so the assumption that $\log|\Sigma| \leq \frac{t}{2^9(\log n)^2} - 2\log n - 6$ must be incorrect; and instead we must have

$$\begin{aligned}
\log|\Sigma| &\geq \frac{t}{2^9(\log n)^2} - 2\log n - 6 \geq \frac{\lfloor r/\lceil \log n + 1\rceil\rfloor}{2^9(\log n)^2} - 2\log n - 6 \\
&= \Omega(r/(\log n)^3 - \log n)\,.
\end{aligned} \tag{8.8}$$

To handle the case of small r, we note that a white-box algorithm $\mathcal{B}$ for $\textsc{mif}(n, r)$ with error $\delta \leq 1/\left(16n\right)^{2\log n+7}$ can be used to solve the $\textsc{avoid}(n, r, 1)$ communication task. Here, Alice, on being given a set $A \subseteq \binom{[n]}{r}$, runs an instance of $\mathcal{B}$ on a sequence containing the elements of A in some order; she then sends the state of the instance to Bob, who queries the instance for an output, and reports that value. This communication protocol has the same error probability as $\mathcal{B}$; by Lemma 3.2, it requires

$$\geq \min\left(\log(r+1), \log\frac{\log 1/\delta}{\log en/r}\right) \geq \log(\min(r+1, 2\log n + 7)) \geq 1$$

bits of communication; thus $\mathcal{B}$ requires at least one bit of state. Since $\max(1, z/a - b) \geq z/(a(1 + b))$, this lets us find a more convenient corollary for Eq. 8.8; that $\log|\Sigma| = \Omega(r/(\log n)^4)$. □

We will now prove Lemma 8.1. It relies on the following technical claim about probability distributions; which *roughly* implies that when a distribution is split into a small number of regions on which it is approximately uniform, a specific sum of powers of the weight and density of each region has a lower bound.

LEMMA 8.2. *Define $m_\phi(K)$ to be the minimum value of distribution ϕ on the set K, so $m_\phi(K) := \min_{i\in K} \phi(i)$.*

Let $p > 1$, $\beta \in (0, 1]$, and $n \geq 2$. For any distribution ν on $[n]$, there exists a collection of disjoint sets $\{H_i\}_{i\in J}$ for some $|J| \leq \frac{3}{\beta}\log n$ where:

$$\sum_{i\in J} (m_\nu(H_i))^{p-1} \frac{(\nu(H_i))^p}{\|\nu\|_p^p} \geq \frac{(1-\frac{1}{n})^p}{2^{\beta(2p-1)}|J|^{(p-1)p/(2p-1)} n^{(p-1)^2/(2p-1)}} \tag{8.9}$$

$$\geq \frac{(1-\frac{1}{n})^p}{2^{\beta(2p-1)}|J|^{(p-1)} n^{(p-1)^2}}\,. \tag{8.10}$$

Furthermore, we have $\max_{i\in J} m_\nu(H_i) \geq 1/(n2^\beta)$, and $\min_{i\in J} m_\nu(H_i) \geq 1/n^2$.

Proof. (Of Lemma 8.1.) In order to avoid awkward expressions like $M(x,R)(i)$, we define $\widetilde{\mu}_x := M(x,R)$. We also use the notation $a^+ := \max(0,a)$. Throughout the proof we shall assume $n \geq 2$, as in the case $n = 1$ it is easy to prove that no such map M exists.

This proof has two main stages. The first establishes that, for a small fraction of vectors x drawn from ν^t, the distribution $\widetilde{\mu}_x$ will probably have significant mass in the same area as ν^t, while not being much more concentrated (according to $\|\cdot\|_p^p$) than $\widetilde{\mu}_x$, and avoids x. The second part will show that such distributions can avoid only small fraction of vectors sampled from ν^t; together, these stages imply the range of M must be large.

Given a real random variable W, with $\mathbb{E}W \geq y$, and $0 \leq W \leq \eta y$, we have

$$\Pr[W \geq \alpha y] = 1 - \Pr[W \leq \alpha y] \geq 1 - \Pr[(\eta y - W) \geq (\eta - \alpha) y] \geq 1 - \frac{\eta y - y}{(\eta - \alpha) y} \geq \frac{1-\alpha}{\eta}\,. \tag{8.11}$$

Let $\beta \in (0,1]$ be a parameter chosen later. Apply Lemma 8.2 to ν with this β and the given p, producing disjoint sets $\{H_i\}_{i \in J}$. For any $i \in J$, we have $\mathbb{E}_{X\sim\nu^t,R}\widetilde{\mu}_X = \nu(H_i)$. Now applying Jensen's inequality to convex functions of the form $f(a) = ((a-b)^+)^p$ gives:

$$\mathbb{E}_{X\sim\nu^t,R}((\widetilde{\mu}_X(H_i) - \delta|H_i|)^+)^p \geq ((\nu(H_i) - \delta|H_i|)^+)^p \qquad \text{which implies}$$
$$\mathbb{E}_{X\sim\nu^t,R}[\sum_{i\in J} m_\nu(H_i)^{p-1}((\widetilde{\mu}_X(H_i) - \delta|H_i|)^+)^p] \geq \sum_{i \in J} m_\nu(H_i)^{p-1}((\nu(H_i) - \delta|H_i|)^+)^p\,.$$

Next, for any $x \in [n]^t$,

$$\sum_{i\in J} m_\nu(H_i)^{p-1}((\widetilde{\mu}_x(H_i) - \delta|H_i|)^+)^p) \leq \max_{i\in J} m_\nu(H_i)^{p-1} \leq (4n)^p \sum_{i\in J} m_\nu(H_i)^{p-1}(\nu(H_i) - \delta|H_i|)^p)\,,$$

because as noted in Lemma 8.2, for the i maximizing $m_\nu(H_i)$, we have $\nu(H_i) \geq |H_i|\frac{1}{n2^\beta} \geq \frac{|H_i|}{2n}$, so $\nu(H_i) - \delta|H_i| \geq |H_i|(\frac{1}{2n} - \frac{1}{n^3}) \geq 1/4n$. Note that $(4n)^p \leq 4^2(n^{1+1/\log n}) = 2^5 n$. Applying Eq. 8.11 thus yields:

$$\Pr_{X\sim\nu^t,R}\left[\begin{array}{l}\displaystyle\sum_{i\in J} m_\nu(H_i)^{p-1}((\widetilde{\mu}_X(H_i) - \delta|H_i|)^+)^p \quad \geq \\ \displaystyle\quad\left(1 - \frac{1}{n}\right)\sum_{i\in J} m_\nu(H_i)^{p-1}((\nu(H_i) - \delta|H_i|)^+)^p\end{array}\right] \geq \frac{1}{n}\cdot\frac{1}{(4n)^p} \geq \frac{1}{2^5 n^2}\,.$$

Intersecting this event with that of Eq. 8.5 implies the probability that all three of the following conditions hold is $\geq \frac{1}{2^6 n^2}$:

$$\begin{aligned}
&(\mathrm{a}): \quad && \|\widetilde{\mu}_X\|_p^p \leq D^{p-1}\|\nu\|_p^p \\
&(\mathrm{b}): \quad && \forall j \in [t] : \widetilde{\mu}_X(X_j) \leq \delta \\
&(\mathrm{c}): \quad && \sum_{i\in J} m_\nu(H_i)^{p-1}((\widetilde{\mu}_X(H_i) - \delta|H_i|)^+)^p \geq (1 - \frac{1}{n})\sum_{i\in J} m_\nu(H_i)^{p-1}((\nu(H_i) - \delta|H_i|)^+)^p\,.
\end{aligned}$$

By the averaging argument, there must exist a value $R' \in \Omega$ for which, when $R = R'$, the above three conditions hold with at least the same probability. In other words, when replacing $\widetilde{\mu}_X$ with $\mu_X := M(x,R')$, the conditions still holds with probability $\geq 1/2^6 n^2$. Now let $G := \{\mu_x : x \in [n]^t \text{ satisfies (a),(c)}\}$ and define $L_\pi := \{i \in [n] : \pi(i) \leq \delta\}$. Therefore,

$$\begin{aligned}
\frac{1}{2^6 n^2} &\leq \Pr_{X\sim\nu^t}[\mu_X \in G \wedge (\forall j \in [t] : \mu_X(X_j) \leq \delta)] \\
&= \sum_{y\in G}\Pr_{X\sim\nu^t}[\mu_X = \mu_y \wedge (\forall j \in [t] : \mu_X(X_j) \leq \delta)] \\
&\leq \sum_{y\in G}\Pr_{X\sim\nu^t}[(\forall j \in [t] : \mu_y(X_j) \leq \delta)] \\
&= \sum_{y\in G}\prod_{j\in[t]}\Pr_{X_j\sim\nu}[\mu_y(X_j) \leq \delta] = \sum_{y\in G}(\nu(L_{\mu_y}))^t
\end{aligned} \tag{8.12}$$

We will now prove an upper bound on $\nu(L_{\mu_y})$ for any given $y \in G$. Observe that for any sequence $a_1, \dots, a_\ell$ of nonnegative real numbers, $\sum_{i=1}^{\ell} a_i^p \geq (\sum_{i=1}^{\ell} a_i)^p / \ell^{p-1}$; this follows from Hölder's inequality. As the sets $H_i \setminus L_{\mu_y}$ are disjoint,

$$\|\mu_y\|_p^p = \sum_{i \in [n]} \mu_y(i)^p \geq \sum_{i \in J} \frac{\mu_y(H_i \setminus L_{\mu_y})^p}{|H_i \setminus L_{\mu_y}|^{p-1}} \, .$$

The definition of L_{μ_y} implies $\mu_y(H_i \setminus L_{\mu_y}) \geq \max(0, \mu_y(H_i) - \delta|H_i|)$. Also, because the minimum value of ν on $H_i \setminus L_{\mu_y}$ is at least $m_\nu(H_i)$, we have

$$|H_i \setminus L_{\mu_y}| \leq \frac{\nu(H_i \setminus L_{\mu_y})}{m_\nu(H_i)} \leq \frac{1 - \nu(L_{\mu_y})}{m_\nu(H_i)}$$

Therefore,

$$\begin{aligned}
\|\mu_y\|_p^p &\geq \sum_{i \in J} \frac{((\mu_y(H_i) - \delta|H_i|)^+)^p}{(1 - \nu(L_{\mu_y}))^{p-1} / m_\nu(H_i)^{p-1}} \\
&= \frac{1}{(1 - \nu(L_{\mu_y}))^{p-1}} \sum_{i \in J} m_\nu(H_i)^{p-1} ((\mu_y(H_i) - \delta|H_i|)^+)^p \\
&\geq \frac{1 - 1/n}{(1 - \nu(L_{\mu_y}))^{p-1}} \sum_{i \in J} m_\nu(H_i)^{p-1} ((\nu(H_i) - \delta|H_i|)^+)^p \qquad \text{by condition (c)} \\
&\geq \frac{1 - 1/n}{(1 - \nu(L_{\mu_y}))^{p-1}} \sum_{i \in J} m_\nu(H_i)^{p-1} \left(\left(1 - \frac{1}{n}\right) \nu(H_i) \right)^p
\end{aligned}$$

The last step uses the fact that for all $i \in J$, $\delta|H_i| \leq \frac{1}{2n^3}|H_i| \leq \frac{1}{n}|H_i| \min_{j \in \bigcup H_i} \nu(j) \leq \frac{1}{n}\nu(H_i)$. We now apply condition (a), and divide both sides by $\|\nu\|_p^p$:

$$\begin{aligned}
D^{p-1} \geq \frac{\|\mu_y\|_p^p}{\|\nu\|_p^p} &\geq \frac{(1 - 1/n)^{p+1}}{(1 - \nu(L_{\mu_y}))^{p-1}} \sum_{i \in J} m_\nu(H_i)^{p-1} \frac{(\nu(H_i))^p}{\|\nu\|_p^p} \\
&\geq \frac{(1 - 1/n)^{p+1}}{(1 - \nu(L_{\mu_y}))^{p-1}} \frac{(1 - 1/n)^p}{2^{\beta(2p-1)} (\frac{2}{\beta} \log n)^{(p-1)} n^{(p-1)^2}} \, . \qquad \text{by Lemma 8.2}
\end{aligned}$$

Rearranging this inequality to isolate $\nu(L_{\mu_y})$ reveals:

$$\nu(L_{\mu_y}) \leq 1 - \frac{1}{D} \left(\frac{(1 - 1/n)^{(2p-1)/(p-1)}}{2^{\beta(2p-1)/(p-1)} (\frac{2}{\beta} \log n) n^{(p-1)}} \right) \tag{8.13}$$

The right hand side is close to its minimum when $\beta = p - 1 = 1/\log n$: Thus:

$$\begin{aligned}
\nu(L_{\mu_y}) &\leq 1 - \frac{1}{D} \left(\frac{(1 - 1/n)^{2 + \log n}}{2^{(2 + \log n)/\log n} (2(\log n)^2) 2^{\log n / \log n}} \right) \\
&\leq 1 - \frac{1}{D} \left(\frac{(1 - 1/n)^{2 + \log n}}{16 \cdot 2^{2/\log n}} \right) \frac{1}{(\log n)^2} \\
&\leq 1 - \frac{1}{2^9 D (\log n)^2}
\end{aligned}$$

since $(1 - 1/n)^{2 + \log n} / (16 \cdot 2^{2/\log n})$ is increasing in n, and when evaluated at $n = 2$ gives 2^{-9}. Now we are in a position to simplify Eq. 8.12; with this upper bound.

$$\frac{1}{2^6 n^2} \leq \sum_{y \in G} \left(1 - \frac{1}{2^9 D (\log n)^2} \right)^t = |G| \left(1 - \frac{1}{2^9 D (\log n)^2} \right)^t \leq |G| \exp\left(- \frac{t}{2^9 D (\log n)^2} \right) \, .$$

Since G is a subset of range(M), we have $\log|\operatorname{range}(M)| \geq \log|G|$; rearranging the above to isolate $|G|$ gives:

$$\begin{aligned}\log|\operatorname{range}(M)| \geq \log|G| &\geq \frac{t}{2^9(\ln 2)D(\log n)^2} - \log(2^6 n^2)\\ &\geq \frac{t}{2^9 D(\log n)^2} - 2\log(n) - 6\,.\end{aligned}$$

□

Finally, we prove Lemma 8.2:

Proof. For all $i \in \mathbb{Z}_{\geq 0}$, let $w_i := 2^{i\beta}/n^2$, and let $H_i := \{j \in [n] : \nu(j) \in [w_i, w_{i+1})$. Define $J := \{i \in \mathbb{Z}_{\geq 0} : H_i \neq \emptyset\}$. We first prove some basic properties of J and the H_i.

- If $i \geq \left\lfloor \frac{3}{\beta}\log n \right\rfloor$, then $w_i \geq 2^{\left\lfloor \frac{3}{\beta}\log n\right\rfloor \beta}/n^2 > 2^{\frac{2}{\beta}(\log n)\beta}/n^2 \geq 1$; since $\max_{j\in[n]}\nu(j) \leq 1$, it follows such H_i must be empty. Thus $J \subseteq \{0, \ldots, \left\lfloor \frac{3}{\beta}\log n \right\rfloor - 1\}$, and hence $|J| \leq \frac{3}{\beta}\log n$.

- Because $\min_{j\in[n]}\nu(j) \geq 1/n$, we are guaranteed that for some i with $w_i \geq 1/(n2^\beta)$, $H_i \neq \emptyset$. Thus $\max_{i\in J} m_\nu(H_i) \geq 1/(n2^\beta)$. Similarly, $\min_{i\in J} m_\nu(H_i) \geq \min_{i\in J} w_i \geq 1/n^2$.

Now, to prove the main part of the result, Eq. 8.10. Let $K = [n] \setminus \bigcup_{i\in J} H_i = \{j \in [n] : \nu(j) < 1/n^2\}$. First, we observe that the contribution of the $j \in K$ to $\|\nu\|_p^p$ is small and can be easily be accounted for:

$$\begin{aligned} n\frac{1}{n^p} \leq \sum_{j\in[n]}\nu(j)^p = \sum_{j\in\bigcup_{i\in J}H_i}\nu(j)^p + \sum_{j\in K}\nu(j)^p &\leq \sum_{j\in\bigcup_{i\in J}H_i}\nu(j)^p + n\left(\frac{1}{n^2}\right)^p\\ \leq \sum_{j\in\bigcup_{i\in J}H_i}\nu(j)^p + \frac{1}{n^p}\sum_{j\in[n]}\nu(j)^p\,. \end{aligned}$$

This implies $\|\nu\|_p^p \leq (1-1/n^p)^{-1}\sum_{j\in\bigcup_{i\in J}H_i}\nu(j)^p \leq (1-1/n)^{-1}\sum_{j\in\bigcup_{i\in J}H_i}\nu(j)^p$.

Now, writing $n_i = |H_i|$, we have $n_i w_i \leq \nu(H_i) \geq 2^\beta n_i w_i$, and so:

$$\sum_{i\in J}(m_\nu(H_i))^{p-1}\frac{(\nu(H_i))^p}{\|\nu\|_p^p} \geq \frac{(1-1/n)\sum_{i\in J}w_i^{p-1}(w_i n_i)^p}{\sum_{i\in J}n_i(2^\beta w_i)^p} = \frac{1-1/n}{2^{\beta p}}\frac{\sum_{i\in J}w_i^{2p-1}n_i^p}{\sum_{i\in J}w_i^p n_i}\,. \tag{8.14}$$

We shall later need the following inequality:

$$\sum_{i\in J}n_i w_i \geq 2^{-\beta}\sum_{j\in\bigcup_{i\in J}H_i}\nu(j) \geq 2^{-\beta}\Big(\sum_{j\in[n]}\nu(j) - \sum_{j\in K}\nu(j)\Big) \geq 2^{-\beta}\left(1-\frac{1}{n}\right)\,. \tag{8.15}$$

With this and properties of the n_i, we can lower bound Eq. 8.14 by using Hölder's inequality several times:

$$\begin{aligned}
\sum_{i\in J} w_i^p n_i &= \sum_{i\in J}\left(w_i^{2p-1}n_i^p\right)^{\frac{p}{2p-1}}\left(n_i^{-(p-1)}\right)^{\frac{p-1}{2p-1}} \\
&\le \left(\sum_{i\in J} w_i^{2p-1}n_i^p\right)^{\frac{p}{2p-1}}\left(\sum_{i\in J} n_i^{-(p-1)}\right)^{\frac{p-1}{2p-1}} && \text{by Hölder}
\end{aligned}$$

$$\le \left(\sum_{i\in J} w_i^{2p-1}n_i^p\right)^{\frac{p}{2p-1}} |J|^{\frac{p-1}{2p-1}} \qquad \text{since } n_i \ge 1 \tag{8.16}$$

$$\begin{aligned}
\sum_{i\in J} w_i n_i &= \sum_{i\in J}\left(w_i^{2p-1}n_i^p\right)^{\frac{1}{2p-1}}\left(n_i^{1/2}\right)^{\frac{2p-2}{2p-1}} \\
&\le \left(\sum_{i\in J} w_i^{2p-1}n_i^p\right)^{\frac{1}{2p-1}}\left(\sum_{i\in J} n_i^{1/2}\right)^{\frac{2p-2}{2p-1}} && \text{by Hölder}\\
&\le \left(\sum_{i\in J} w_i^{2p-1}n_i^p\right)^{\frac{1}{2p-1}}\left(\left(\sum_{i\in J} n_i\right)^{1/2}|J|^{1/2}\right)^{\frac{2p-2}{2p-1}} && \text{by Cauchy-Schwarz}
\end{aligned}$$

$$\le \left(\sum_{i\in J} w_i^{2p-1}n_i^p\right)^{\frac{1}{2p-1}} n^{\frac{p-1}{2p-1}}|J|^{\frac{p-1}{2p-1}} \qquad \text{since } \sum_{i\in J} n_i \le n. \tag{8.17}$$

Multiplying Eq. 8.16 by Eq. 8.17 raised to the $(p-1)$st power gives:

$$\left(\sum_{i\in J} w_i^p n_i\right)\left(\sum_{i\in J} w_i n_i\right)^{p-1} \le \left(\sum_{i\in J} w_i^{2p-1}n_i^p\right)|J|^{\frac{p-1}{2p-1}}(n^{\frac{p-1}{2p-1}})^{p-1}(|J|^{\frac{p-1}{2p-1}})^{p-1},$$

which implies

$$\frac{\sum_{i\in J} w_i^{2p-1}n_i^p}{\sum_{i\in J} w_i^p n_i} \ge \frac{\left(\sum_{i\in J} w_i n_i\right)^{p-1}}{|J|^{\frac{(p-1)p}{2p-1}} n^{\frac{(p-1)^2}{2p-1}}} \ge \frac{(1-\frac{1}{n})^{p-1}}{2^{\beta(p-1)}|J|^{\frac{(p-1)p}{2p-1}} n^{\frac{(p-1)^2}{2p-1}}}.$$

Substituting this into Eq. 8.14 gives:

$$\sum_{i\in J}(m_\nu(H_i))^{p-1}\frac{(\nu(H_i))^p}{\|\nu\|_p^p} \ge \frac{(1-\frac{1}{n})^p}{2^{\beta(2p-1)}|J|^{\frac{(p-1)p}{2p-1}} n^{\frac{(p-1)^2}{2p-1}}}.$$

□

9 Random start and pseudo-deterministic models

THEOREM 9.1. *The space needed for an algorithm in the random-start model to solve* $\textsc{mif}(n,r)$ *against adaptive adversaries, with error* $\le \delta \le \frac{1}{6}$*, satisfies* $s \ge S^{PD}_{1/3}(n, \lfloor r/(2s+2)\rfloor)$.

This theorem implies that *if* it is the case that $S^{PD}_{1/3}(\textsc{mif}(n,r)) = \Omega(r^c/\operatorname{polylog}(n))$ for some constant $c > 0$, then it follows that $S^{RS}_{1/6}(\textsc{mif}(n,r)) = \Omega(r^{c/(1+c)}/\operatorname{polylog} n)$. Specifically, if $s = S^{RS}_{1/6}(\textsc{mif}(n,r))$, then Theorem 9.1 would imply $s \ge S^{PD}_{1/3}(n, \lfloor r/(2s+2)\rfloor) = \Omega((r/s)^c/\operatorname{polylog}(n))$; multiplying both sides by s^c and raising them to the $1/(c+1)$st power gives $s \ge \Omega(r^{c/(1+c)}/\operatorname{polylog}(n))$.

Proof. Let Σ be the set of all states of the random-start algorithm $\mathcal{A}$, and let $\mathcal{D}$ be the distribution of the initial states of the algorithm. Write $B \sim \mathcal{A}$ to indicate that B is an instance of $\mathcal{A}$, i.e., with initial state drawn from distribution $\mathcal{D}$. Let $\ell = 2\lceil \log(|\Sigma|) \rceil + 2$, and let $t = \lfloor r/\ell \rfloor$. For any partial stream σ of elements, and instance B of $\mathcal{A}$, we let $B(\sigma)$ be the sequence of $|\sigma|$ outputs made by B after it processes each element in σ.

Consider an adversary E which does the following. Given σ the stream it has already passed to the algorithm, and ω the sequence of outputs that $\mathcal{A}$ produced in response to σ, the adversary checks if there exists any $x \in [n]^t$ for which

$$\forall y \in [n]^t : \Pr_{B\sim\mathcal{A}}[B(\sigma.x) = \omega.y \mid B(\sigma) = \omega] \leq \frac{2}{3}\,. \tag{9.18}$$

If so, it sends x to $\mathcal{A}$, appends x to σ and the returned t elements to ω, and repeats the process. If no such x exists, then the adversary identifies the $z \in [n]^t$ which maximizes:

$$\Pr_{B\sim\mathcal{A}}[B(\sigma.z) \text{ is incorrect} \mid B(\sigma) = \omega]\,. \tag{9.19}$$

and sends it to the algorithm. (The adversary gives up if either the algorithm manages to give a valid output after z, or after it has sent ℓ sets of t elements to the algorithm.)

We claim that if $\log|\Sigma| < S^{PD}_{1/3}(\textsc{mif}(n,t))$, then E makes the algorithm fail with probability $\geq 1/6$. There are two ways that E can be forced to give up: if it tries more than $\ell - 1$ times to find a point where there is no $x \in [n]^t$ satisfying Eq. 9.18, or if the z it sends fails to produce an error.

Assume that the adversary finds a value of x satisfying Eq. 9.18, for each of the ℓ tries it makes. Let $x_1, \ldots, x_\ell \in [n]^t$ be these values, and let $y_1, \ldots, y_\ell \in [n]^t$ be the outputs of the algorithm. By applying Eq. 9.18 repeatedly, we have:

$$\begin{aligned}
\Pr_{B\sim\mathcal{A}}[B(x_1.\ldots.x_\ell) = y_1.\ldots.y_\ell] &= \Pr_{B\sim\mathcal{A}}[B(x_1.\ldots.x_\ell) = y_1.\ldots.y_\ell \mid B(x_1.\ldots.x_{\ell-1}) = y_1.\ldots.y_{\ell-1}] \\
&\quad \cdot \Pr_{B\sim\mathcal{A}}[B(x_1.\ldots.x_{\ell-1}) = y_1.\ldots.y_{\ell-1} \mid B(x_1.\ldots.x_{\ell-2}) = y_1.\ldots.y_{\ell-2}] \\
&\quad \cdot \Pr_{B\sim\mathcal{A}}[B(x_1) = y_1] \\
&\leq (2/3)^\ell\,.
\end{aligned}$$

Let $C \subseteq \Sigma$ be the set of initial states of the algorithm for which the adversary finds a sequence satisfying Eq. 9.18, ℓ times. Because the algorithm is deterministic after the initial state is chosen, each $s \in C$ has a corresponding transcript $(\sigma_s, \omega_s) \in [n]^{t\ell} \times [n]^{t\ell}$ that occurs when E is run against an instance of $\mathcal{A}$ started from s. Therefore,

$$\begin{aligned}
\Pr_{s\sim\mathcal{D}}[s \in C] &= \sum_{s\in C} \Pr_{s'\sim cD}[s = s'] \leq \sum_{s\in C} \Pr_{B\sim\mathcal{A}}[B(\sigma_s) = \omega_s] \\
&\leq |\Sigma| \left(\frac{2}{3}\right)^\ell \leq 2^{\log|\Sigma|} \left(\frac{2}{3}\right)^{2\log(|\Sigma|)+2} \leq \left(\frac{2}{3}\right)^2 \leq \frac{1}{2}
\end{aligned}$$

Thus, the chance that E fails to find a point where no x satisfying Eq. 9.18 exists is $\leq \frac{1}{2}$.

To bound the second way in which E can fail, we let (σ, ω) be a partial transcript of the algorithm for which no $x \in [n]^t$ satisfies Eq. 9.18. Assume the probability that z produces an error is $< 1/3$. Then we have:

$$\forall z \in [n]^t, \exists y_z \in [n]^t : \qquad \Pr_{B\sim\mathcal{A}}[B(\sigma.z) = \omega.y_z \mid B(\sigma) = \omega] \geq \frac{2}{3} \tag{9.20}$$

$$\forall z \in [n]^t : \qquad \Pr_{B\sim\mathcal{A}}[B(\sigma.z) \text{ is correct} \mid B(\sigma) = \omega] \geq \frac{2}{3}\,. \tag{9.21}$$

These conditions together imply that $\omega.y_z$ is a correct $\textsc{mif}(n,r)$ output sequence for $\sigma.z$. As a result, we can use $\mathcal{A}$'s behavior after (σ, ω) to construct a pseudo-deterministic algorithm Ψ for $MIF(n,t)$. To initialize Ψ, we sample an initial state $B \sim \mathcal{A}$ conditioned on the event that $B(\sigma) = \omega$, and then send the elements of σ to B. After this, when Ψ receives an element e, we send e to B, and report the element B outputs as the output of Ψ. By Eqs. 9.20 and 9.21, the sequence of outputs produced by Ψ on any input x in $[n]^t$ will, with probability $\geq 2/3$, be the (valid) output y_z. Thus, Ψ solves $MIF(n,t)$ with $\leq 1/3$ error – which, under the assumption that $\log|\Sigma| < S^{PD}_{1/3}(\textsc{mif}(n,t))$, is impossible. Thus the z chosen by the adversary makes the algorithm err with probability $\geq 1/3$, conditional on it having found (σ, ω) with no $x \in [n]^t$ satisfying Eq. 9.18. The probability that the the adversary succeeds is then $\geq 1/3 \cdot 1/2 = 1/6$; this contradicts the assumption that $\mathcal{A}$ has error $\leq 1/6$ against any adversary, which implies that we must instead have $\log|\Sigma| \geq S^{PD}_{1/3}(\textsc{mif}(n,t))$. □

Acknowledgements

We thank Amit Chakrabarti and Prantar Ghosh for many helpful discussions.

References

[1] M. Ajtai, , V. Braverman, T. Jayram, S. Silwal, A. Sun, D. P. Woodruff, and S. Zhou, *The white-box adversarial data stream model*, in Proc. 41st ACM Symposium on Principles of Database Systems, 2022, p. 15–27, `https://doi.org/10.1145/3517804.3526228`.

[2] S. Assadi, A. Chen, and G. Sun, *Deterministic graph coloring in the streaming model*, in Proc. 54th Annual ACM Symposium on the Theory of Computing, 2022, pp. 261—-274, `https://doi.org/10.1145/3519935.3520016`.

[3] S. Assadi, Y. Chen, and S. Khanna, *Sublinear algorithms for (Δ+ 1) vertex coloring*, in Proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms, 2019, pp. 767–786, `https://doi.org/10.1137/1.9781611975482.48`.

[4] O. Ben-Eliezer, R. Jayaram, D. P. Woodruff, and E. Yogev, *A framework for adversarially robust streaming algorithms*, in Proc. 39th ACM Symposium on Principles of Database Systems, 2020, p. 63–80, `https://doi.org/10.1145/3375395.3387658`.

[5] O. Ben-Eliezer and E. Yogev, *The adversarial robustness of sampling*, in Proc. 39th ACM Symposium on Principles of Database Systems, ACM, 2020, pp. 49–62, `https://doi.org/10.1145/3375395.3387643`.

[6] A. Chakrabarti, P. Ghosh, and M. Stoeckl, *Adversarially robust coloring for graph streams*, in Proc. 13th Conference on Innovations in Theoretical Computer Science, 2022, pp. 37:1–37:23, `https://doi.org/10.4230/LIPIcs.ITCS.2022.37`.

[7] U. Feige, *A randomized strategy in the mirror game*, (2019), `https://arxiv.org/abs/1901.07809`.

[8] S. Garg and J. Schneider, *The Space Complexity of Mirror Games*, in Proc. 10th Conference on Innovations in Theoretical Computer Science, 2018, pp. 36:1–36:14, `https://doi.org/10.4230/LIPIcs.ITCS.2019.36`.

[9] S. Goldwasser, O. Grossman, S. Mohanty, and D. P. Woodruff, *Pseudo-Deterministic Streaming*, in Proc. 20th Conference on Innovations in Theoretical Computer Science, vol. 151, 2020, pp. 79:1–79:25, `https://doi.org/10.4230/LIPIcs.ITCS.2020.79`.

[10] S. Goldwasser, R. Impagliazzo, T. Pitassi, and R. Santhanam, *On the pseudo-deterministic query complexity of NP search problems*, in Proc. 36th Annual IEEE Conference on Computational Complexity, 2021, pp. 36:1–36:22, `https://doi.org/10.4230/LIPIcs.CCC.2021.36`.

[11] M. Hardt and D. P. Woodruff, *How robust are linear sketches to adaptive inputs?*, in Proc. 45th Annual ACM Symposium on the Theory of Computing, 2013, pp. 121–130, `https://doi.org/10.1145/2488608.2488624`.

[12] A. Hassidim, H. Kaplan, Y. Mansour, Y. Matias, and U. Stemmer, *Adversarially robust streaming algorithms via differential privacy*, in Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020, `https://proceedings.neurips.cc/paper/2020/hash/0172d289da48c48de8c5ebf3de9f7ee1-Abstract.html`.

[13] A. Hatcher, *Algebraic Topology*, Cambridge University Press, 2002. Available online at `https://pi.math.cornell.edu/~hatcher/AT/ATpage.html`. Accessed 2022-07-14.

[14] H. Jowhari, M. Saglam, and G. Tardos, *Tight bounds for l_p samplers, finding duplicates in streams, and related problems*, in Proc. 30th ACM Symposium on Principles of Database Systems, 2011, pp. 49–58, `https://doi.org/10.1145/1989284.1989289`.

[15] H. Kaplan, Y. Mansour, K. Nissim, and U. Stemmer, *Separating adaptive streaming from oblivious streaming using the bounded storage model*, in Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III, vol. 12827 of Lecture Notes in Computer Science, Springer, 2021, pp. 94–121, `https://doi.org/10.1007/978-3-030-84252-9_4`.

[16] M. Kapralov, J. Nelson, J. Pachocki, Z. Wang, D. P. Woodruff, and M. Yahyazadeh, *Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams*, in Proc. 58th Annual IEEE Symposium on Foundations of Computer Science, 2017, pp. 475–486, `https://doi.org/10.1109/FOCS.2017.50`.

[17] B. Menuhin and M. Naor, *Keep that card in mind: Card guessing with limited memory*, in Proc. 13th Conference on Innovations in Theoretical Computer Science, 2022, pp. 107:1–107:28, `https://doi.org/10.4230/LIPIcs.ITCS.2022.107`.

[18] S. Muthukrishnan, *Data streams: Algorithms and applications*, Found. Trends Theor. Comput. Sci., 1 (2005), pp. 117–236, `http://dx.doi.org/10.1561/0400000002`.

[19] M. Naor and E. Yogev, *Bloom filters in adversarial environments*, ACM Trans. Alg., 15 (2019), pp. 35:1–35:30, `https://doi.org/10.1145/3306193`.

[20] J. Tarui, *Finding a duplicate and a missing item in a stream*, in Proc. 4th International Conference on Theory and Applications of Models of Computation, 2007, pp. 128–135, `https://doi.org/10.1007/978-3-540-72504-6_11`.

[21] D. P. Woodruff and S. Zhou, *Tight bounds for adversarially robust streams and sliding windows via difference estimators*, in Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science, 2022, pp. 1183–1196, https://doi.org/10.1109/FOCS52979.2021.00116.

A Appendix

Proof. In this proof of Lemma 3.3, we essentially repeat the proof of the Chernoff bound, with slight modifications to account for the dependence of X_i on its predecessors. Here t is a positive real number chosen later.

$$
\begin{aligned}
&\Pr\left[\sum_{i=1}^{n} X_i \geq np(1+\delta)\right] \\
&\quad = \Pr\left[e^{t\sum_{i=1}^{n} X_i} \geq e^{tnp(1+\delta)}\right] \\
&\quad \leq \mathbb{E}[e^{t\sum_{i=1}^{n} X_i}]/e^{tnp(1+\delta)} \\
&\quad = e^{-tnp(1+\delta)}\mathbb{E}[e^{tX_1}\mathbb{E}[e^{tX_2}\cdots\mathbb{E}[e^{tX_n} \mid X_1 = X_1, \ldots, X_{n-1} = X_{n-1}] \mid X_1 = X_1] \\
&\quad \leq e^{-tnp(1+\delta)}\mathbb{E}[e^{tX_1}\mathbb{E}[e^{tX_2}\cdots\mathbb{E}[e^{tX_{n-1}}(pe^t + (1-p)) \mid X_1 = X_1, \ldots, X_{n-2} = X_{n-2}] \mid X_1 = X_1] \\
&\quad \leq e^{-tnp(1+\delta)}\mathbb{E}[e^{tX_1}\mathbb{E}[e^{tX_2}\cdots\mathbb{E}[e^{tX_{n-2}}(pe^t + (1-p))^2 \mid X_1 = X_1, \ldots, X_{n-3} = X_{n-3}] \mid X_1 = X_1] \\
&\quad \leq e^{-tnp(1+\delta)}(pe^t + (1-p))^n = \left(\frac{pe^t + (1-p)}{e^{tp(1+\delta)}}\right)^n \\
&\quad \leq \left(\frac{e^{p(e^t-1)}}{e^{tp(1+\delta)}}\right)^n = \left(\frac{e^{e^t-1}}{e^{t(1+\delta)}}\right)^{np} \qquad \text{since } 1+x \leq e^x \\
&\quad = \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{np} \qquad \text{picking } t = \ln(1+\delta) \\
&\quad \leq \exp\left(-\frac{\delta^2 np}{2+\delta}\right). \qquad \text{since } x - (1+x)\ln(1+x) \leq -x^2/(2+x)
\end{aligned}
$$

□