Finding missing items requires strong forms of randomness*

Amit Chakrabarti† Manuel Stoeckl†

Abstract

Adversarially robust streaming algorithms are required to process a stream of elements and produce correct outputs, even when each stream element can be chosen as a function of earlier algorithm outputs. As with classic streaming algorithms, which must only be correct for the worst-case fixed stream, adversarially robust algorithms with access to randomness can use significantly less space than deterministic algorithms. We prove that for the Missing Item Finding problem in streaming, the space complexity also significantly depends on how adversarially robust algorithms are permitted to use randomness. (In contrast, the space complexity of classic streaming algorithms does not depend as strongly on the way randomness is used.)

For Missing Item Finding on streams of length ℓ with elements in $\{1,\ldots,n\}$, and $\leq 1/\text{poly}(\ell)$ error, we show that when $\ell = O(2^{\sqrt{\log n}})$, "random seed" adversarially robust algorithms, which only use randomness at initialization, require $\ell^{\Omega(1)}$ bits of space, while "random tape" adversarially robust algorithms, which may make random decisions at any time, may use $O(\text{polylog}(\ell))$ space. When ℓ is between $n^{\Omega(1)}$ and $O(\sqrt{n})$, "random tape" adversarially robust algorithms need $\ell^{\Omega(1)}$ space, while "random oracle" adversarially robust algorithms, which can read from a long random string for free, may use $O(\text{polylog}(\ell))$ space. The space lower bound for the "random seed" case follows, by a reduction given in prior work, from a lower bound for pseudo-deterministic streaming algorithms given in this paper.

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

†Department of Computer Science, Dartmouth College, Hanover NH 03755, USA.

1 Introduction

Randomized streaming algorithms can achieve exponentially better space bounds than corresponding deterministic ones: this is a basic, well-known, easily proved fact that applies to a host of problems of practical interest. A prominent class of randomized streaming algorithms uses randomness in a very specific way, namely to sketch the input stream by applying a random linear transformation—given by a sketch matrix S—to the input frequency vector. The primary goal of a streaming algorithm is to achieve sublinear space, so it is infeasible to store S explicitly. In some well-known cases, the most natural presentation of the algorithm is to explicitly describe the distribution of S, a classic case in point being frequency moment estimation [Ind06]. This leads to an algorithm that is very space-efficient *provided one doesn't charge the algorithm any space cost for storing* S. Algorithms that work this way can be thought of as accessing a "random oracle": despite their impracticality, they have theoretical value, because the standard ways of proving space *lower* bounds for randomized streaming algorithms in fact work in this model. For the specific frequency-moment algorithms mentioned earlier, [Ind06] goes on to design variants of his algorithms that use only a small (sublinear) number of random bits and apply a pseudorandom generator to suitably mimic the behavior of his random-oracle algorithms. Thus, at least in this case, a random *oracle* isn't necessary to achieve sublinear complexity. This raises a natural question: from a space complexity viewpoint, does it ever help to use a random oracle, as opposed to "ordinary" random bits that must be stored (and thus paid for) if they are to be reused?

For most classic streaming problems, the answer is "No," but for unsatisfactory reasons: Newman's Theorem [New91] allows one to replace a long oracle-provided random string by a much shorter one (that is cheap to store), though the resulting algorithm is non-constructive. This brings us to the recent and ongoing line of work on *adversarially robust* streaming algorithms where we shall find that the answer to our question is a very interesting "Yes." For the basic and natural MISSINGITEMFINDING problem, defined below, we shall show that three different approaches to randomization result in distinct space-complexity behaviors. To explain this better, let us review adversarial robustness briefly.

Some recent works have studied streaming algorithms in a setting where the input to the algorithm can be adaptively (and adversarially) chosen based on its past outputs. Existing ("classic") randomized streaming algorithms may fail in this *adversarial setting* when the input-generating adversary learns enough about the past random choices of the algorithm to identify future inputs on which the algorithm will likely fail. There are, heuristically, two ways for algorithm designers to protect against this: (a) prevent the adversary from learning the past random choices of the algorithm (in the extreme, by making a pseudo-deterministic algorithm), or (b) prevent the adversary from exploiting knowledge of past random decisions, by having the algorithm's future behavior depend on randomness that it has not yet revealed. Concretely, algorithms in this setting use techniques such as independent re-sampling [BY20], sketch switching using independent sub-instances of an underlying classic algorithm [BJWY20], rounding outputs to limit the number of computation paths [BJWY20], and differential privacy to safely aggregate classic algorithm sub-instances [HKM+20]. Mostly, these algorithms use at most as many random bits as their space bounds allow. However, some recently published adversarially robust streaming algorithms for vertex-coloring a graph (given by an edge stream) [CGS22, ACGS23], and one for the MISSINGITEMFINDING problem [Sto23], assume access to a large amount of oracle randomness: they prevent the adversary from exploiting the random bits it learns by making each output depend on an unrevealed part of the oracle random string. It is still open whether these last two problems have efficient solutions that do not use this oracle randomness hammer. This suggests the following question:

Are there problems for which space-efficient adversarially robust streaming algorithms provably require access to oracle randomness?

In this paper, we prove that for certain parameter regimes, MISSINGITEMFINDING (henceforth, MIF)

is such a problem. In the problem MIF(n,ℓ), the input is a stream $\langle e_1,\ldots,e_\ell\rangle$ of ℓ integers, not necessarily distinct, with each $e_i \in \{1,\ldots,n\}$, where $1 \le \ell \le n$. The goal is as follows: having received the ith integer, output a number v in $\{1,\ldots,n\} \setminus \{e_1,\ldots,e_i\}$. We will be mostly interested in the setting $\ell = o(n)$, so the "trivial" upper bound on the space complexity of MIF(n,ℓ) is $O(\ell \log n)$, achieved by the deterministic algorithm that simply stores the input stream as is.

1.1 Groundwork for Our Results

To state our results about MIF, we need to introduce some key terminology. Notice that MIF is a *tracking problem*: an output is required after reading each input.[1] Thus, we view streaming algorithms as generalizations of finite state (Moore-type) machines. An algorithm $\mathcal{A}$ has a finite set of states Σ (leading to a space cost of $\log_2|\Sigma|$), a finite input set $\mathcal{I}$, and a finite output set $\mathcal{O}$. It has a transition function $T : \Sigma \times \mathcal{I} \times \mathcal{R} \to \Sigma$ indicating the state to switch to after receiving an input, plus an output function γ: $\Sigma \times \mathcal{R} \to \mathcal{O}$ indicating the output produced upon reaching a state. How the final parameter (in $\mathcal{R}$) of T and γ is used depends on the type of randomness. We consider four cases, leading to four different models of streaming computation.

- *Deterministic.* The initial state of the algorithm is a fixed element of Σ, and T and γ are deterministic (they do not depend on the parameter in $\mathcal{R}$).
- *Random seed.* The initial state is drawn from a distribution $\mathcal{D}$ over Σ, and T and γ are deterministic. This models the situation that all random bits used count towards the algorithm's space cost.
- *Random tape.* The initial state is drawn from a distribution $\mathcal{D}$ over Σ.[2] The space $\mathcal{R}$ is a sample space; when the algorithm receives an input $e \in \mathcal{I}$ and is at state $\sigma \in \Sigma$, it chooses a random $\rho \in \mathcal{R}$ independent of all previous choices and moves to state $T(e,\sigma,\rho)$. However, γ is deterministic.[3] This models the situation that the algorithm can make random decisions at any time, but it cannot remember past random decisions without recording them (which would add to its space cost).
- *Random oracle.* The initial state is fixed; $\mathcal{R}$ is a sample space. A specific $R \in \mathcal{R}$ is drawn at the start of the algorithm and stays the same over its lifetime. When the algorithm is at state σ and receives input e, its next state is $T(e,\sigma,R)$. The output given at state σ is $\gamma(\sigma,R)$. This models the situation that random bits are essentially "free" to the algorithm; it can read from a long random string which doesn't count toward its space cost and which remains consistent over its lifetime. A random oracle algorithm can be interpreted as choosing a random deterministic algorithm, indexed by R, from some family.

These models form a rough hierarchy; they have been presented in (almost) increasing order of power. Every z-bit (2^z-state) deterministic algorithm can be implemented in any of the random models using z bits of space; the same holds for any z-bit random seed algorithm. Every z-bit random tape algorithm has a corresponding $(z+\log\ell)$-bit random oracle algorithm—the added space cost is because for a random oracle algorithm to emulate a random tape algorithm, it must have a way to get "fresh" randomness on each turn.[4]

Streaming algorithms are also classified by the kind of correctness guarantee they provide. Recall that we focus on "tracking" algorithms [BJWY20]; they present an output after reading each input item and this

[1] We do not consider algorithms with a "one-shot" guarantee, to only be correct at the end of the stream, because a) the adversarial setting requires tracking output b) for MIF and most other problems the difference in space complexity is generally small.

[2] Requiring that this model use a fixed initial state could make some algorithms use one additional "INIT" state.

[3] Alternatively, we could associate a *distribution* of outputs to each state, or a function mapping (input, state) pairs to outputs. As these formulations are slightly more complicated to prove things with, and only affect the space usage of MISSINGITEMFINDING algorithms by an additive $O(\log n + \log\frac{1}{\delta})$ amount, we stick with the one state = one output convention.

[4] An alternative, which lets one express z-bit random tape algorithms using a z-bit random oracle variant, is to assume the random oracle algorithm has access to a clock or knows the position in the stream for free; both are reasonable assumptions in practice.

entire sequence of outputs must be correct. Here are three possible meanings of the statement "algorithm $\mathcal{A}$ is δ-error" (we assume that $\mathcal{A}$ handles streams of length ℓ with elements in $\mathcal{I}$ and has outputs in $\mathcal{O}$):

- *Static setting.* For all inputs $\tau \in \mathcal{I}^\ell$, running $\mathcal{A}$ on τ produces incorrect output with probability $\leq \delta$.
- *Adversarial setting.* For all (computationally unbounded) adaptive adversaries α (i.e., for all functions $\alpha\colon \mathcal{O}^\star \to \mathcal{I}$),[5] running $\mathcal{A}$ against α will produce incorrect output with probability $\leq \delta$.
- *Pseudo-deterministic setting.* There exists a canonical output function $f\colon \mathcal{I}^\star \to \mathcal{O}$ producing all correct outputs so that, for each $\tau \in \mathcal{I}^\ell$, $\mathcal{A}(\tau)$ fails to output $f(\tau)$ with probability $\leq \delta$.

Algorithms for the static setting are called "classic" streaming algorithms; ones for the adversarial setting are called "adversarially robust" streaming algorithms. All pseudo-deterministic algorithms are adversarially robust, and all adversarially robust algorithms are also classic.

As a consequence of Newman's theorem [New91], any random oracle or random tape algorithm in the static setting with error δ can be emulated using a random seed algorithm with only ε increase in error and an additional $O(\log \ell + \log\log |\mathcal{I}| + \log \frac{1}{\varepsilon\delta})$ bits of space. However, the resulting algorithm is non-constructive.

1.2 Our Results

As context for our results, we remind the reader that it's trivial to solve $\textsc{mif}(n,\ell)$ in $O(\ell \log n)$ space deterministically (somewhat better deterministic bounds were obtained in [Sto23]). Moving to randomized algorithms, [Sto23] gave a space bound of $O(\log^2 n)$ for $\ell \leq n/2$ in the static setting, and a bound of $\widetilde{O}(\ell^2/n+1)$ [6] in the adversarial setting, using a random *oracle*. The immediate takeaway is that, given access to a deep pool of randomness (i.e., an oracle), MIF becomes easy in the static setting for essentially the full range of stream lengths ℓ and remains easy even against an adversary for lengths $\ell \leq \sqrt{n}$.

The main results of this paper consist of three new lower bounds and one new upper bound on the space complexity of $\textsc{mif}(n,\ell)$. Stating the bounds in their strongest forms leads to complicated expressions; therefore, we first present some easier-to-read takeaways from these bounds that carry important conceptual messages. In the lower bounds below, the error level should be thought of as $\delta = 1/n^2$.

Result 1. At $\ell = \sqrt{n}$, adversarially robust random tape algorithms for $\textsc{mif}(n,\ell)$ require $\Omega(\ell^{1/4})$ bits of space. More generally, for every constant $\alpha \in (0,1)$, there is a constant $\beta \in (0,1)$ such that at $\ell = \Omega(n^\alpha)$, the space requirement is $\Omega(\ell^\beta)$, in the adversarially robust random tape setting.

This shows that MIF remains hard, even for modest values of ℓ, if we must be robust while using only a random *tape*, i.e., if there is a cost to storing random bits we want to reuse—a very reasonable requirement for a practical algorithm. The above result is an exponential separation between the random tape and random oracle models.

The random *seed* model places an even greater restriction on an algorithm: besides counting towards storage cost, random bits are available only at initialization and not on the fly. Many actual randomized algorithms, including streaming ones, are structured this way, making it a natural model to study. We obtain the following result.

Result 2. Adversarially robust random seed algorithms for $\textsc{mif}(n,\ell)$ require $\widetilde{\Omega}(\sqrt{\ell})$ bits of space.

Consider the two results above as ℓ decreases from $\sqrt{n}$ to $\Theta(1)$. The bound in Result 2 stays interesting even when $\ell = n^{o(1)}$, so long as $\ell \geq (\log n)^C$ for a suitable constant C (in fact, the full version of the result is

[5] By the minimax theorem, it suffices to consider deterministic adversaries.

[6] The notations $\widetilde{O}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ hide factors polylogarithmic in n and ℓ.

Setting	Type	Bound	Reference
Static	Random seed	$O((\log n)^2)$ if $\ell \le n/2$	[Sto23][a]
Adversarial	Random oracle	$O((\frac{\ell^2}{n} + \log n)\log n)$	[Sto23]
		$\Omega(\frac{\ell^2}{n})$	[Sto23]
Adversarial	Random tape	$O(\ell^{\log_n \ell}(\log \ell)^2 + \log \ell \cdot \log n)$ †	Theorem 5.6
		$\Omega(\ell^{\frac{15}{32}\log_n \ell})$ †	Theorem 4.8
Adversarial	Random seed	$O((\frac{\ell^2}{n} + \sqrt{\ell} + \log n)\log n)$	[Sto23][b]
		$\Omega(\frac{\ell^2}{n} + \sqrt{\frac{\ell}{(\log n)^3}} + \ell^{1/5})$	Theorem 6.11
Pseudo-deterministic	Random oracle	$\Omega(\frac{\ell}{(\log(2n/\ell))^2} + (\ell \log n)^{1/4})$	Theorem 6.9
Static	Deterministic	$\Omega(\frac{\ell}{\log(2n/\ell)} + \sqrt{\ell})$	[Sto23]
		$O(\frac{\ell \log \ell}{\log n} + \sqrt{\ell \log \ell})$	[Sto23]

Table 1: Bounds for the space complexity of MIF(n,ℓ), from this and prior work. To keep expressions simple, these bounds are evaluated at error level $\delta = 1/n^2$, when applicable. (†) indicates that the precise results are stronger.

[a] This is obtained by accounting for the randomness cost of [Sto23]'s random *oracle* algorithm for the static setting.
[b] The random seed algorithm for the adversarial setting is given in the arXiv version of [Sto23].

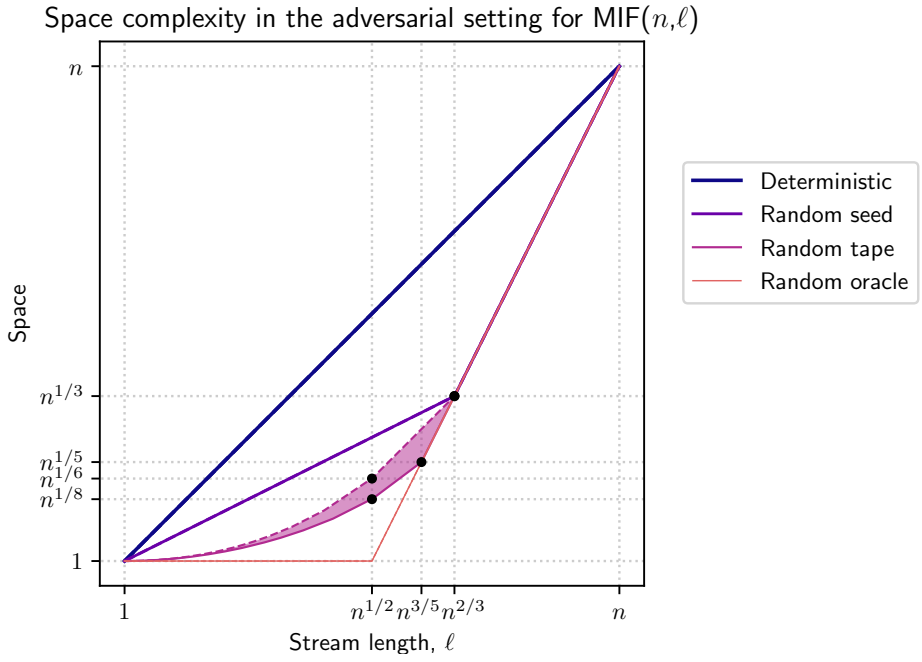

Figure 1: Known bounds for the space complexity of MIF(n,ℓ) in different streaming models, at error level $\delta = 1/n^2$. This is a log-log plot. Upper and lower bounds are drawn using lines of the same color; the region between them is shaded. The upper and lower bounds shown all match (up to polylog(n) factors) except for the case of adversarially robust, random tape algorithms. Pseudo-deterministic and deterministic complexities match within polylog(n) factors.

good for even smaller ℓ). In contrast, the bound in Result 1 peters out at much larger values of ℓ. There is a very good reason: MIF starts to become "easy," even under a random-tape restriction, once ℓ decreases to sub-polynomial in n. Specifically, we obtain the following *upper* bound.

Result 3. There is an adversarially robust random tape algorithm for MIF(n,ℓ) that, in the regime $\ell = O(2^{\sqrt{\log n}})$, uses $O(\log \ell \cdot \log n)$ bits of space.

Notice that at $\ell = \Theta(2^{(\log n)^{1/C}})$, where $C \geq 2$ is a constant, the bound in Result 3 is polylogarithmic in ℓ. Combined with the lower bound in Result 2, we have another exponential separation, between the random seed and random tape models.

Empirically, the existing literature on streaming algorithms consists of numerous linear-sketch-based algorithms, which tend to be efficient in the random seed model, and sampling-based algorithms, which naturally fit the (stronger) random tape model. In view of this, the combination of Results 2 and 3 carries the following important message.

Sampling is provably more powerful than sketching in an adversarially robust setting.

The proof of Result 2 uses a reduction, given in prior work [Sto23], that converts a space lower bound in the pseudo-deterministic setting to a related bound in the random-seed setting. A pseudo-deterministic algorithm is allowed to use randomness (which, due to Newman's theorem, might as well be of the oracle kind) but must, with high probability, map each input to a *fixed* output, just as a deterministic algorithm would. This strong property makes the algorithm adversarially robust, because the adversary has nothing to learn from observing its outputs. Thanks to the [Sto23] reduction, the main action in the proof of Result 2 is the following new lower bound we give.

Result 4. Pseudo-deterministic random oracle algorithms for MIF(n,ℓ) require $\widetilde{\Omega}(\ell)$ bits of space.

These separations rule out the possibility of a way to convert an adversarially robust random oracle algorithm to use only a random *seed* or even a random *tape*, with only minor (e.g., a polylog(ℓ,n) factor) overhead. In contrast, as we noted earlier, such a conversion is routine in the static setting, due to Newman's theorem [New91]. The separation between random oracle and random tape settings shows that MISSINGITEMFINDING is a problem for which much lower space usage is possible if one's adversaries are computationally bounded (in which case a pseudo-random generator can emulate a random oracle.)

Table 1 shows more detailed versions of the above results as well as salient results from earlier work. Together with Figure 1, it summarizes the state of the art for the space complexity of MIF(n,ℓ). The fully detailed versions of our results, showing the dependence of the bounds on the error probability, appear in later sections of the paper, as indicated in the table.

1.3 Related Work

We briefly survey related work. An influential early work [HW13] considered adaptive adversaries for *linear* sketches. The adversarial setting was formally introduced by [BJWY20], who provided general methods (like sketch-switching) for designing adversarially robust algorithms given classic streaming algorithms, especially in cases where the problem is to approximate a real-valued quantity. For some tasks, like F_0-estimation, they obtained slightly better upper bounds by using a random oracle, although later work [WZ22] removed this need. [BY20] observed that in sampling-based streaming algorithms, increasing the sample size is often all that is needed to make an algorithm adversarially robust. [HKM+20] described how to use differential privacy techniques as a more efficient alternative to sketch-switching, and [BEEO22] used this as part of a more efficient adversarially robust algorithm for turnstile F_2-estimation.

Most of these papers focus on providing algorithms and general techniques, but there has been some work on proving adversarially robust lower bounds. [KMNS21] described a problem (of approximating a certain real-valued function) that requires exponentially more space in the adversarial setting than in the static setting. [CGS22], in a brief comment, observed a similar separation for a simple problem along the lines of MIF. They also proved lower bounds for adversarially robust coloring algorithms for graph edge-insertion streams. [Sto23] considered the MIF problem as defined here and, among upper and lower bounds in a number of models, described an adversarially robust algorithm for MIF that requires a random oracle; they asked whether a random oracle is *necessary* for space-efficient algorithms.

The *white-box* adversarial setting [ABJ+22] is similar to the adversarial setting we study, with the adversary having the additional power of seeing the internal state of the algorithm, including (if used) the random oracle. [Sto23] proved an $\Omega(\ell/\text{polylog}(n))$ lower bound for $\text{MIF}(n,\ell)$ for random tape algorithms in this setting, suggesting that any more efficient algorithm for MIF must conceal some part of its internal state. Pseudo-deterministic streaming algorithms were introduced by [GGMW20], who gave lower bounds for a few problems. [BKKS23, GGS23] gave lower bounds for pseudo-deterministic algorithms that approximately count the number of stream elements. The latter shows they require $\Omega(\log m)$ space, where m is the stream length; in contrast, in the static setting, Morris's counter algorithm[7] uses only $O(\log\log m)$ space.

While it is not posed as a streaming task, the *mirror game* introduced by [GS18] is another problem with conjectured separation between the space needed for different types of randomness. In the mirror game, two players (Alice and Bob) alternately state numbers in the set $\{1,\ldots,n\}$, where n is even, without repeating any number, until one player mistakenly states a number said before (loss) or the set is completed (tie). [GS18] showed that if Alice has $o(n)$ bits of memory and plays a deterministic strategy, Bob can always win. Later, [Fei19, MN22b] showed that if Alice has access to a random oracle, she can tie-or-win w.h.p. using only $O(\text{polylog}(n))$ space. A major open question here is how much space Alice needs when she does not have a random oracle. [MN22a] did not resolve this, but showed that if Alice is "open-book" (equivalently, that Bob is a white-box adversary and can see her state), then Alice needs $\Omega(n)$ bits of state to tie-or-win.

Assuming access to a random oracle is a reasonable temporary measure when designing streaming algorithms in the static setting. As noted at the beginning of Section 1, [Ind06] designed L_p-estimation algorithms using random linear sketch matrices, without regard to the amount of randomness used, and then described a way to apply Nisan's PRG [Nis90] to partially derandomize these algorithms and obtain efficient (random seed) streaming algorithms. In general, the use of PRGs for linear sketches has some space overhead, which later work (see [JW23] as a recent example) has been working to eliminate.

It is important to distinguish the "random oracle" type of streaming algorithm from the "random oracle model" in cryptography [BR93], in which one assumes that *all* agents have access to the random oracle. [ABJ+22], when defining white-box adversaries, also assumed that they can see the same random oracle as the algorithm; and, for one task, obtained a more efficient algorithm against a computationally bounded white-box adversary, when both have access to a random oracle, than when neither do. Tight lower bounds are known in neither case.

The power of different types of access to randomness has been studied in computational complexity. [Nis93] showed that logspace Turing machines with a multiple-access random tape can (with zero error) decide languages that logspace Turing machines with a read-once random tape decide only with bounded two-sided error. This type of separation does not hold for *time* complexity classes.

For a more detailed history and survey of problems related to MISSINGITEMFINDING, we direct the reader to [Sto23].

[7] Morris's is a "random tape" algorithm; "random seed" algorithms for counting aren't better than deterministic ones.

2 Technical Overview

The proofs of Results 1, 3 and 4 are all significant generalizations of existing proofs from [Sto23] which handled different (and more tractable) models. The proof of Result 2 consists of applying a reduction from [Sto23] to the lower bound given by Result 4. As we explain our techniques, we will summarize the relevant "basic" proofs from [Sto23], which will clarify the enhancements needed to obtain our results.

Space complexity lower bounds in streaming models are often proved via communication complexity. This meta-technique is unavailable to us, because the setup of communication complexity blurs the distinctions between random seed, random tape, and random oracle models and our results are all about these distinctions. Instead, to prove Result 1, we design a suitable strategy for the stream-generating adversary that exploits the algorithm's random-tape limitation by learning enough about its internal state. Our adversary uses a nontrivially recursive construction. To properly appreciate it, it is important to understand what streaming-algorithmic techniques the adversary must contend with. Therefore, we shall discuss our *upper* bound result first.

2.1 Random Tape Upper Bound (Result 3; Theorem 5.6)

The adversarially robust random tape algorithm for $\textsc{mif}(n,\ell)$ can be seen as a generalization of the random oracle and random seed algorithms.

The random oracle algorithm and its adversaries. The random oracle algorithm for $\textsc{mif}(n,\ell)$ from [Sto23] has the following structure. It interprets its oracle random string as a uniformly random sequence L containing $\ell+1$ distinct elements in $[n]$. As it reads its input, it keeps track of which elements in L were in the input stream so far (were "covered"). It reports as its output the first uncovered element of L. Because L comes from the oracle, the space cost of the algorithm is just the cost of keeping track of the set J of covered positions in L. We will explain why that can be done using only $O((\ell^2/n+1)\log\ell)$ space, in expectation.

An adversary for the algorithm only has two reasonable strategies for choosing the next input. It can "echo" back the current algorithm output to be the next input to the algorithm. It can also choose the next input to be a value from the set U of values that are neither an earlier input nor the current output—but because L is chosen uniformly at random, one can show that the adversary can do no better than picking the next input uniformly at random from U. (The third strategy, of choosing an old input, has no effect on the algorithm.) When the algorithm is run against an adversary that chooses inputs using a mixture of the echo and random strategies, the set J will be structured as the union of a contiguous interval starting at 1 (corresponding to the positions in L covered by the echo strategy) and a sparse random set of expected size $O(\ell^2/n)$ (corresponding to positions in L covered by the random strategy). Together, these parts of J can be encoded using $O((\ell^2/n+1)\log\ell)$ bits, in expectation.

Delaying the echo strategy. If we implemented the above random oracle algorithm as a random seed algorithm, we would need $\Omega(\ell)$ bits of space, just to store the random list L. But why does L need to have length $\ell+1$? This length is needed for the algorithm to be resilient to the echo strategy, which covers one new element of L on every step; if L were shorter, the echo strategy could entirely cover it, making the algorithm run out of possible values to output. The random seed algorithm for $\textsc{mif}(n,\ell)$ works by making the echo strategy less effective, ensuring that multiple inputs are needed for it to cover another element of L. It does this by partitioning $[n]$ into $\Theta(\ell)$ disjoint subsets ("blocks") of size $\Theta(n/\ell)$, and then taking L to be a random list of blocks (rather than a random list of elements of $[n]$). We will now say that a block is "covered" if *any* element of that block was an input. Instead of outputting the first uncovered element in L, the algorithm will run a deterministic algorithm for MIF *inside* the block corresponding to the first uncovered block of L, and report outputs from that; and will only move on to the next uncovered block when the nested algorithm stops. See Listing 1 for the details of this design. Because the analogue of the echo

strategy now requires many more inputs to cover a block, we can make the list L shorter. This change will not make the random strategy much more effective.[8] The minimum length of L is constrained by the $O(n/\ell)$ block sizes, which limit the number of outputs that the nested algorithm can make; as a result, one must have $L = \Omega(\ell^2/n)$. In the end, after balancing the length of the list with the cost of the nested algorithm, the optimal list length for the random seed algorithm will be $O(\ell^2/n + \sqrt{\ell})$.

Listing 1 Example: recursive construction for a random tape MIF(n,ℓ) algorithm, building on algorithm $\mathcal{A}$

Parameter: $t \in [\Omega(\ell^2/n), \ell]$ is the number of parts into which the input stream is split

Initialization:

Let $k = O(t)$, $s = O(\ell)$, and $B_1, \ldots, B_s$ be a partition of $[n]$ into s equal "blocks" ▷ *assuming $s \mid n$*
$L \leftarrow$ uniformly randomly chosen sequence of k distinct elements of $[s]$
$J \leftarrow \varnothing$, is a subset of $[k]$ ▷ *a set marking which blocks of L have been covered*
$c \leftarrow 1$ ▷ *the current active block*
$A \leftarrow$ instance of algorithm $\mathcal{A}$ solving MIF$(n/s, \lceil \ell/t \rceil)$

Update$(a \in [n])$:

Let h be the block containing a, and x the rank of a in B_h
if $h \in L$ **then**
    Add j to J, where $L_j = h$ ▷ *Mark list element containing h as used*
if $h = L_c$ **then**
    A.UPDATE(x)
if A is out of space **then** ▷ *This requires that A.UPDATE() be called $\geq \lceil \ell/t \rceil$ times*
    $c \leftarrow$ least integer which is $> c$ and not in J ▷ *This line may abort if $J = [k]$*
    $A \leftarrow$ new instance of algorithm $\mathcal{A}$ ▷ *Using new random bits, if $\mathcal{A}$ is randomized*

Output $\rightarrow [n]$:

Let $x \in [n/s]$ be the output of A
return xth entry of block B_c

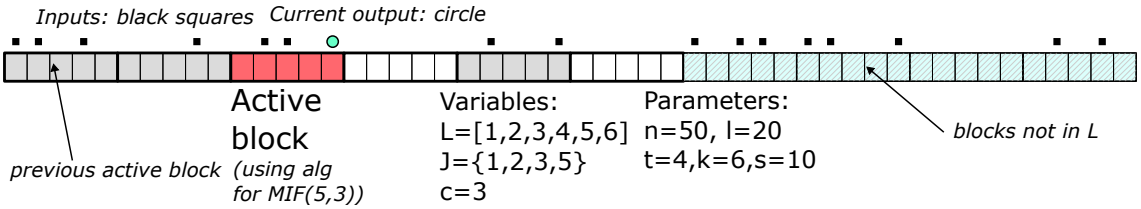

Figure 2: A diagram illustrating the state of an instance of Listing 1 on an example input. Positions on the horizontal axis correspond to integers in $[n]$; the set of values in the input stream ($\{1,2,4,9,12,13,\ldots\}$) is marked with black squares; the current output value (15) with a circle. Outside this example, L need not be contiguous or in sorted order.

The recursive random tape algorithm. The random seed algorithm for MIF(n,ℓ) used the construction of Listing 1 to build on top of an "inner" deterministic algorithm.[9] To get an efficient random tape algorithm, we can recursively apply the construction of Listing 1 $d-1$ times, for $d = O(\min(\log \ell, \log n/\log \ell))$; at

[8] The fact that $[n]$ is split into $\Omega(\ell)$ blocks is enough to mitigate the random strategy; with ℓ guesses, the adversary is unlikely to guess more than a constant fraction of the elements in L.

[9] The construction uses randomness in two places: when initializing the random sequence L, and (possibly) each time the inner algorithm is initialized. For the random seed model, every "inner" initialization would require a corresponding set of random bits, which are counted toward the space cost of the algorithm. Using a deterministic inner algorithm avoids this cost.

the end of this recursion, we can use a simple deterministic algorithm for MIF. The optimal lengths of the random lists used at each level of the recursion are determined by balancing the costs of the different recursion levels. We end up choosing list lengths that all bounded by a quantity which lies between $O(\ell^{1/d})$ and $O(\ell^{1/(d-1)})$.

In the extreme case where $d = \Theta(\log \ell)$ and the required error level δ is constant, our recursive algorithm may have a stack of random lists, each of length 2, and every time a level of the algorithm completes (i.e., all blocks of a list have been used), it will make a new instance of that level. That is, some large uncovered block will be split into many smaller blocks, and the algorithm will randomly pick two of them for the new instance's list. Because the lists are all short, the algorithm will not need to remember many random bits at a given time; in exchange, for this regime it needs a very large ($n = \ell^{\Omega(d)}$) number of possible outputs and will frequently need to sample new random lists.

We defer the exact implementation details to the final version of our algorithm, Listing 3. It looks somewhat different from the recursive construction in Listing 1, because we have unraveled the recursive framing to allow for a simpler error analysis that must only bound the probability of a single "bad event."

2.2 Random Tape Lower Bound (Result 1; Theorem 4.8)

The AVOID problem. At the core of many of the MIF lower bounds is the SUBSETAVOIDANCE communication problem, introduced in [CGS22]. Here we have two players, Alice and Bob, and a known universe $[m]$: Alice has a set $A \subseteq [m]$ of size a, and should send a message (as short as possible) to Bob, who should use the message to output a set $B \subseteq [m]$ of size b which is disjoint from A. Henceforth, we'll call this problem AVOID(m,a,b). [CGS22] showed that both deterministic and constant-error randomized one-way protocols for this problem require $\Omega(ab/m)$ bits of communication. An adversarially robust z-space algorithm for MIF$(m, a+b)$ can be used as a subroutine to implement a z-bit one-way protocol for AVOID(m,a,b), thereby proving $z = \Omega(ab/m)$. This immediately gives us an $\Omega(\ell^2/n)$ space lower bound for MIF(n,ℓ), which, as we have seen, is near-optimal in the robust, random oracle setting.

The random tape lower bound. To prove stronger lower bounds that exploit the random tape limitation of the algorithm, we need a more sophisticated use of AVOID. Fix an adversarially robust, random tape, z-space algorithm $\mathcal{A}$ for MIF(n,ℓ). Roughly speaking, while the random oracle argument used $\mathcal{A}$ to produce an AVOID protocol at the particular scale $a = b = \ell$, for the fixed universe $[n]$, our random tape argument will "probe" $\mathcal{A}$ in a recursive fashion—reminiscent of the recursion in our random tape upper bound—to identify a suitable scale and sub-universe at which an AVOID protocol can be produced. This probing will itself invoke the AVOID lower bound to say that if an AVOID(m,a,b) protocol is built out of a z-space streaming algorithm where $z \ll a$, then B must be small, with size $b = O((z/a)m)$.

We will focus on the regime where $\delta = O(1/n)$. This error level requires a measure of structure from the algorithm: it cannot just pick a random output each step, because that would risk colliding with an earlier input with $\geq 1/n$ probability. Our recursive argument works by writing z, the space usage of $\mathcal{A}$, as a function of a space lower bound for MIF(w,t), where $w = \Theta(zn/\ell)$ and $t = \Theta(\ell/z)$. For small enough z, $t^2/w \gg \ell^2/n$, so by repeating this reduction step a few times we can increase the ratio of the stream length to the input domain size until we can apply the simple $\Omega(\hat{\ell}^2/\hat{n})$ lower bound for MIF$(\hat{n},\hat{\ell})$. With the right number of reduction steps, one obtains the lower bound formula of Theorem 4.8, of which Result 1 is a special case.

The reduction. The reduction step argues that the MIF(n,ℓ) algorithm $\mathcal{A}$ "contains" a z-space algorithm for MIF(w,t), which, on being given any $t = O(\ell/z)$ items in a certain sub-universe $W \subseteq [n]$ of size $w = O(zn/\ell)$, will repeatedly produce missing items *from that sub-universe*. That such a set W *exists* can be seen as a consequence of the lower bound for AVOID: if $\mathcal{A}$ receives a random sorted subset S of $\ell/2$ elements in $[n]$, then because there are $\binom{n}{\ell/2}$ possible subsets, most of the 2^z states of $\mathcal{A}$ will need to be "good" for $\Omega(2^{-z}\binom{n}{\ell/2})$ different subsets. In particular, upon reaching a given state σ, for $\mathcal{A}$ to solve MIF with error

probability $O(1/n)$, its outputs henceforth—for the next $\ell/2$ items in the stream—must avoid most of the sets of inputs that could have led it to σ. We will prove by a counting argument (Lemma 4.4) that after the random sequence S is sent, each state σ has an associated set H_σ of possible "safe" outputs which are unlikely to collide with the inputs from S, and that $|H_\sigma|$ is typically $O(zn/\ell)$. Thus, for a typical state σ, starting $\mathcal{A}$ from σ causes its next $\ell/2$ outputs to be inside H_σ, w.h.p.; in other words, $\mathcal{A}$ contains a "sub-algorithm" solving $\text{MIF}(O(zn/\ell), \ell/2)$ on the set $W = H_\sigma$.

However, even though there exists a set W on which $\mathcal{A}$ will concentrate its outputs, it may not be possible for an adversary to find it. In particular, had $\mathcal{A}$ been a random *oracle* algorithm, each setting of the random string might lead to a different value for W, making W practically unguessable. But $\mathcal{A}$ is in fact a random *tape* algorithm, so we can execute the following strategy.

In our core lemma, Lemma 4.3, we design an adversary (Listing 2) that can with $\Omega(1)$ probability identify a set W of size $\Theta(zn/\ell)$ for which the next $\Theta(\ell/z)$ outputs of $\mathcal{A}$ will be contained in W, with $\Omega(1)$ probability, no matter what inputs the adversary sends next. In other words, our adversary will identify a part of the stream and a sub-universe of $[n]$ where the algorithm solves $\text{MIF}(\Theta(zn/\ell), \Theta(\ell/z))$. The general strategy is to use an iterative search based on a win-win argument. First, the adversary will send a stream comprising a random subset S of size $\ell/2$ to $\mathcal{A}$, to ensure that henceforth its outputs are contained in some (unknown) set H_ρ, where ρ is the (unknown) state reached by $\mathcal{A}$ just after processing S. Because $\mathcal{A}$ has $\leq 2^z$ states, from the adversary's perspective there are $\leq 2^z$ possible candidates for H_ρ. Then, the adversary conceptually divides the rest of the stream to be fed to $\mathcal{A}$ into $O(z)$ phases, each consisting of $t = O(\ell/z)$ stream items. In each phase, one of the following things happens.

1. There exists a "sub-adversary" (function to choose the t items constituting the phase, one by one) which will probably make $\mathcal{A}$ output an item that rules out a constant fraction of the candidate values for H_ρ (output i rules out set J if $i \notin J$). The adversary then runs this sub-adversary.

2. No matter how the adversary picks the t inputs for this phase, there will be a set W (roughly, an "average" of the remaining candidate sets) that probably contains the corresponding t outputs of $\mathcal{A}$.

As the set of candidate sets can only shrink by a constant fraction $O(z)$ times, the first case can only happen $O(z)$ times, with high probability. Thus, eventually, the adversary will identify the set W that it seeks. Once it has done so, it will run the optimal adversary for $\text{MIF}(\Theta(zn/\ell), \Theta(\ell/z))$. This essentially reduces the lower bound for $\text{MIF}(\ell, n)$ to that for $\text{MIF}(\Theta(zn/\ell), \Theta(\ell/z))$.

One subtlety is that we will need to carefully account for the probability that $\mathcal{A}$, over the next $\Theta(\ell/z)$ stream items, produces outputs outside W. This will require us to distinguish between two types of "errors" for the algorithm over those next $\Theta(\ell/z)$ items: an $O(1)$ chance of producing an output outside W, and a smaller chance of making a mistake per the definition of MIF, i.e., outputting an item that was not missing (cf. Definition 4.1).

2.3 Random Seed Lower Bound via Pseudo-Determinism (Result 2; Theorem 6.11)

The adversary constructed above for our random tape lower bound can be seen as a significant generalization of the adversary used by [Sto23] to prove a random seed lower bound conditioned on a (then conjectured) pseudo-deterministic lower bound. Indeed, [Sto23]'s adversary against a z-space algorithm $\mathcal{A}$ also proceeds in a number of phases, each of length $t = \Theta(\ell/z)$. In each step, either (1) it can learn some new information about the initial state of $\mathcal{A}$ (the "random seed"), by sending $\mathcal{A}$ a specific stream of inputs in $[n]^t$, looking at the resulting output, and ruling out the seed values that could not have produced the output; or (2) it cannot learn much information, because for any possible input stream in $[n]^t$, $\mathcal{A}$ has an output that it produces with constant probability. Each time the adversary follows the case (1), a constant fraction of the $\leq 2^z$ seed values are ruled out. Therefore, either within $O(z)$ steps the adversary will exactly learn the seed, at which point

it can perfectly predict $\mathcal{A}$'s behavior, which lands us in case (2); or $\mathcal{A}$ will not reveal much information about the seed in a given phase, which also puts us in case (2). Because case (2) means that $\mathcal{A}$ behaves pseudo-deterministically, $\mathcal{A}$ must use enough space to pseudo-deterministically solve $\text{MIF}(n,t)$.

Thus, Result 2 follows as a corollary of Result 4, which we discuss next.

2.4 Pseudo-Deterministic Lower Bound (Result 4; Theorem 6.9)

This proof generalizes [Sto23]'s space lower bound for *deterministic* $\text{MIF}(n,\ell)$ algorithms, which we briefly explain. Fix a deterministic $\text{MIF}(n,\ell)$ algorithm $\mathcal{A}$ that uses z bits of space. For each stream τ with length $|\tau| \le \ell$, define F_τ to be the set of *all possible outputs* of $\mathcal{A}$ corresponding to length-ℓ streams that have τ as a prefix. Let ρ be a stream such that $|\tau|+|\rho| \le \ell$. Then, by definition, $F_{\tau\circ\rho} \subseteq F_\tau$ whereas, by the correctness of $\mathcal{A}$, $F_{\tau\circ\rho} \cap \rho = \varnothing$. Now consider the AVOID problem over the universe F_τ, for a fixed τ: if Alice gets $\rho \subseteq F_\tau$ as an input, she could send Bob the state σ of $\mathcal{A}$ upon processing $\tau\circ\rho$, whereupon Bob could determine $F_{\tau\circ\rho}$ (by repeatedly running $\mathcal{A}$'s state machine starting at σ), which would be a valid output.

Let us restrict this scenario to suffixes ρ of some fixed length t; we'll soon determine a useful value for t. By the above observations, were it the case that

$$\exists \tau \in [n]^{\le \ell - t}\ \forall \rho \in [n]^t \colon |F_{\tau\circ\rho}| \ge \tfrac{1}{2}|F_\tau|\,, \tag{1}$$

we would have a z-bit protocol for $\text{AVOID}(|F_\tau|, t, \frac{1}{2}|F_\tau|)$. By the [CGS22] lower bound, we would have $z \ge Ct$ for a universal constant C. On the other hand, if the opposite were true, i.e.,

$$\forall \tau \in [n]^{\le \ell - t}\ \exists \rho \in [n]^t \colon |F_{\tau\circ\rho}| < \tfrac{1}{2}|F_\tau|\,, \tag{2}$$

then, starting from the empty stream ϵ, we could add a sequence of length-t suffixes $\rho_1,\dots,\rho_d$ (where $d \le \lfloor \ell/t \rfloor$) such that $|F_{\rho_1\circ\cdots\circ\rho_d}| < 2^{-d}|F_\epsilon| \le 2^{-d}n$. Since $\mathcal{A}$ must produce *some* output at time ℓ, this would be a contradiction for $d \ge \log n$. Thus, for a setting of $t = \Theta(\ell/\log n)$, situation (1) must occur, implying a lower bound of $z = \Omega(\ell/\log n)$.

Relaxing "all outputs" to "common outputs". Examining the above argument closely shows where it fails for pseudo-deterministic algorithms. In constructing an AVOID protocol above, we needed the key property that F_τ can be determined from just the *state* of $\mathcal{A}$ upon processing τ. For pseudo-deterministic algorithms, if we simply define F'_τ to be "the set of all *canonical* outputs at time ℓ for continuations of τ," we cannot carry out the above proof plan because this F'_τ cannot be computed reliably from a single state: given a random state σ associated to τ, on average a δ fraction of the outputs might be incorrect and have arbitrary values; even a single bad output could corrupt the union calculation!

To work around this issue, we replace F_τ with a more elaborate recursive procedure FINDCOMMONOUTPUTS, (or FCO for short) that computes the "most common outputs" at time ℓ for a certain distribution over continuations of τ. To explain this, let us imagine positions 1 through ℓ in the input stream as being divided into d contiguous "time intervals." In the deterministic proof, these intervals were of length t each. Given a stream τ that occupies the first $d-k$ of these intervals, F_τ can be thought of as the output of a procedure FINDALLOUTPUTS (or FAO for short) where $\text{FAO}(\mathcal{A},\tau,k)$ operates as follows: for each setting ρ of the $(d-k+1)$th time interval, call $\text{FAO}(\mathcal{A},\tau\circ\rho,k-1)$ and return the union of the sets so obtained. In the base case, $\text{FAO}(\mathcal{A},\tau,0)$ takes a stream $\tau \in [n]^\ell$ and returns the singleton set $\{\mathcal{A}(\tau)\}$. The deterministic argument amounts to showing that, with interval lengths $t = \Theta(z)$, the set $\text{FAO}(\mathcal{A},\tau,k)$ has cardinality $\ge 2^k$; since $\text{FAO}(\mathcal{A},\epsilon,d)$ has cardinality $\le n$, this bounds $d \le \log n$, which lower-bounds z.

For our pseudo-deterministic setting, we use time intervals as above and we design an analogous procedure $\text{FCO}(B,C,\tau,k)$ that operates on a function $B\colon [n]^\ell \to [n]$ (roughly corresponding to an MIF algorithm), a matrix C of random thresholds,[10] and a stream τ of length $\le \ell$ that occupies the first $d-k$ time intervals.

[10] The use of random thresholds is a standard trick for robustly computing quantities in the presence of noise.

The recursive structure of $\text{FCO}(B,C,\tau,k)$ is similar to FAO, but crucially, the sets computed by the recursive calls $\text{FCO}(B,C,\tau\circ\rho,k-1)$ are used differently. Instead of simply returning their union, we use these sets to collect statistics about the outputs in $[n]$ and return only those that are sufficiently common. The thresholds in C control the meaning of "sufficiently common."

The function B provided to FCO can be either the canonical output function Π of the given pseudo-deterministic algorithm $\mathcal{B}$ or a deterministic algorithm $A \sim \mathcal{B}$ obtained by fixing the random coins of $\mathcal{B}$. We will show that:

- With high probability over C and the randomness of $\mathcal{B}$, FCO will produce the same outputs on Π and $\mathcal{B}$. In other words, FCO is robust to noise (i.e., to algorithm errors).
- When applied to the canonical algorithm, the cardinalities of the sets returned by FCO will grow exponentially with k. Equivalently, similar to $|F_\tau|$ from the deterministic proof, the cardinality of $\text{FCO}(\tau,\ldots)$ will shrink exponentially as the length $|\tau|$ grows. Ultimately, this is proven by implementing AVOID using FCO on the actual algorithm as a subroutine. Critically, this implementation uses the fact that the recursive calls to FCO w.h.p. produce the same output on Π and $\mathcal{B}$.
- The argument can be carried out with all but one of the d time intervals being of length $\approx \Theta(z)$. If z were too small, d would be large enough that for the empty stream prefix we would have $|\text{FCO}(\epsilon,\ldots)| > n$, which contradicts $\text{FCO}(\ldots) \subseteq [n]$; this lets us derive a lower bound on z.

Error amplification and the case $n \gg \ell$. One technical issue that arises is that the correctness of FCO requires $\mathcal{B}$'s error probability to be as small as $1/n^{\Omega(\log n)}$. Fortunately, even if the original error probability was $1/3$, we can reduce it to the required level since pseudo-deterministic algorithms allow efficient error reduction by independent repetition. A second technical point is that a z-space pseudo-deterministic algorithm can be shown to have only $O(2^z)$ possible outputs; so if $n \gg \ell$, we can sometimes obtain a stronger lower bound by pretending that n is actually $O(2^z)$. This is formalized by a simple encoding argument.

3 Preliminaries

Notation. Throughout this paper, $\log x = \log_2 x$, while $\ln x = \log_e x$. The set $\mathbb{N}$ consists of all positive integers; $[k] := \{1,2,\ldots,k\}$; and $[a,b)$ is a half open interval of real numbers. For a condition or event E, the symbol $\mathbb{1}_E$ takes the value 1 if E occurs and 0 otherwise. The sequence (stream) obtained by concatenating sequences a and b, in that order, is denoted $a \circ b$. For a set S of elements in a totally ordered universe, $\text{SORT}(S)$ denotes the sequence of elements of S in increasing order; $\binom{S}{k}$ is the set of k-element subsets of S; and $\text{SEQS}(S,k) = \{\text{SORT}(Y) : Y \in \binom{S}{k}\}$. We sometimes extend set-theoretic notation to vectors and sequences; e.g., for $y \in [n]^t$, write $y \subseteq S$ to mean that $\forall i \in [t] : y_i \in S$. For a set X, $\triangle[X]$ denotes the set of probability distributions over X, while $A \in_R X$ indicates that A is chosen uniformly at random from X. When naming probability distributions, we will use either calligraphic letters (e.g., $\mathcal{A},\mathcal{D}$) or the letter μ; $A \sim \mu$ means that A is drawn from the distribution μ.

3.1 Useful Lemmas

These will be used in following sections. When no external work is cited, a proof is given for completeness either here or in Appendix A.1.

Lemma 3.1 (Multiplicative Azuma's inequality)**.** *Let $X_1,\dots,X_t$ be $[0,1]$ random variables, and $\alpha \geq 0$. If, for all $i \in [t]$, $\mathbb{E}[X_i \mid X_1,\dots,X_{i-1}] \leq p_i$, then*

$$\Pr\left[\sum_{i=1}^{t} X_i \geq (1+\alpha)\sum_{i=1}^{t} p_i\right] \leq \exp\left(-((1+\alpha)\ln(1+\alpha)-\alpha)\sum_{i=1}^{t} p_i\right) \leq \exp\left(-\frac{\alpha^2}{2+\alpha}\sum_{i=1}^{t} p_i\right).$$

On the other hand, if for all i, $\mathbb{E}[X_i \mid X_1,\dots,X_{i-1}] \geq p_i$, then

$$\Pr\left[\sum_{i=1}^{t} X_i \leq (1-\alpha)\sum_{i=1}^{t} p_i\right] \leq \exp\left(-((1-\alpha)\ln(1-\alpha)+\alpha)\sum_{i=1}^{t} p_i\right) \leq \exp\left(-\frac{\alpha^2}{2}\sum_{i=1}^{t} p_i\right).$$

In contrast to the above, the "usual" form of Azuma's inequality uses a martingale presentation and gives an additive-type bound.

Lemma 3.2 (Chernoff bound with negative association, from [JP83])**.** *The standard multiplicative Chernoff bounds work with negatively associated random variables.*

Lemma 3.3 (Error amplification by majority vote)**.** *Let $\varepsilon \leq \delta \leq 1/3$. Say X is a random variable, and v a value with $\Pr[X = v] \geq 1-\delta$. If $X_1,\dots,X_p$ are independent copies of X, then the most common value in $(X_1,\dots,X_p)$ will be v with probability $\geq 1-\varepsilon$, for $\varepsilon = (2\delta)^{p/30}$.*

The above is a standard lemma, useful for trading error for space for algorithms with a single valid output.

As outlined in Section 2.2, $\textsc{avoid}(m,a,b)$ is a one-way communication problem, wherein player Alice has a set $A \in \binom{[m]}{a}$, and should send a short message to player Bob, who should use the message to output a set $B \in \binom{[m]}{b}$ that is disjoint from A. In the randomized δ-error setting, this disjointness should hold with probability $\geq 1-\delta$.

Theorem 3.4 ($\textsc{avoid}$ communication lower bound, from [CGS22])**.** *Suppose there exists a randomized protocol for $\textsc{avoid}(m,a,b)$, in which Alice communicates $\leq K$ bits, that is δ-error either on a worst-case input or when Alice's input is chosen uniformly at random from $\binom{[t]}{a}$. Then*

$$K \geq \frac{ab}{t\ln 2} + \log(1-\delta).$$

Using Theorem 3.4, it is straightforward to derive the following lower bound for robust algorithms for $\textsc{mif}(n,\ell)$, as was done in [Sto23]. The proof is short, yet instructive, so we outline it here.

Theorem 3.5 (Adversarially robust random oracle lower bound, from [Sto23])**.** *If there exists a δ-error adversarially robust random oracle algorithm for $\textsc{mif}(n,\ell)$ using z bits of space, then*

$$z \geq \frac{\ell^2}{4n\ln 2} + \log(1-\delta).$$

Proof. Such an algorithm $\mathcal{A}$ yields a z-bit δ-error randomized protocol for $\textsc{avoid}(n,\lceil \ell/2\rceil, \lfloor \ell/2\rfloor + 1)$ wherein Alice feeds her set A into $\mathcal{A}$, sends the state of $\mathcal{A}$ to Bob, and Bob extracts set B by using the "echo" adversarial strategy, i.e., repeatedly asking $\mathcal{A}$ for an output item and feeding that item back as the next input. The result now follows by appealing to Theorem 3.4. □

We also note the following simple lower bound.

Lemma 3.6. *For every $\delta < 1$, if there exists a δ-error random tape algorithm for $\textsc{mif}(n,\ell)$ using z bits of space, then $z \geq \log(\ell+1)$.*

Proof. Each state of a random tape streaming algorithm $\mathcal{A}$ has a unique associated output value. If $z < \log(\ell+1)$, then $\mathcal{A}$ has at most ℓ states. Let H be the set of outputs associated with these states; so $|H| \leq \ell$. When sent a stream containing each element of H, $\mathcal{A}$ will fail with probability 1 because every output it could make is wrong. □

By the remarks in Section 1.1 following the definitions of the models of computation, Theorem 3.5 also applies to random tape and random seed algorithms and Lemma 3.6 also applies to random seed algorithms.

4 The Random Tape Lower Bound

This section presents our first and perhaps most important lower bound, of which Result 1 is a consequence. We shall carry out the proof plan outlined in Section 2.2, designing a recursive adversary to foil a given random-tape MIF algorithm $\mathcal{A}$ that runs in z bits of space. Correspondingly, our lower bound proof will be inductive.

4.1 Setup and Base Case

Recall that the adversary organizes the ℓ-length input to be fed into $\mathcal{A}$ as a random prefix of length $\ell/2$ followed by another $\ell/2$ inputs divided into several phases, each consisting of $t = O(\ell/z)$ inputs. The adversary's eventual goal is to identify a particular phase and a corresponding sub-universe $W \subseteq [n]$ so that $\mathcal{A}$, when suitably conditioned and restricted to that phase, yields a sub-algorithm $\mathcal{B}$ that is good for MIF for inputs from W. However, we will need to generalize the notion of a "good" MIF algorithm, because this sub-algorithm might produce outputs outside of W, even when fed inputs from W. To aid our analysis, we will make $\mathcal{B}$ abort anytime it would have produced an output outside of W. In what follows, it will be important to maintain a distinction between these aborts and actual mistakes.

Definition 4.1. An algorithm $\mathcal{A}$ for $\textsc{mif}(n,\ell)$ can fail in either of two ways. It may make an incorrect output, or *mistake*, if outputs an element in $[n]$ that *is* in its input stream (i.e., not missing). It may also *abort*, by outputting a special value $\perp$ (where $\perp \notin [n]$) and stopping its run.

For integers n, ℓ, z with $1 \leq \ell < n$, and $\gamma \in [0,1]$, let $\textsc{Algs}(n,\ell,\gamma,z)$ be the set of all z-bit *random tape* algorithms for $\textsc{mif}(n,\ell)$ which on *any* adversary abort with probability $\leq \gamma$. Define

$$\Delta(n,\ell,\gamma,z) := \min_{\mathcal{A} \in \textsc{Algs}(n,\ell,\gamma,z)} \delta_{\max}(\mathcal{A},n,\ell)\,,$$

where $\delta_{\max}(\mathcal{A},n,\ell)$ is the maximum probability, over all possible adversaries, that $\mathcal{A}$ makes a mistake. As a consequence of the definition, $\Delta(n,\ell,\gamma,z)$ is non-increasing in γ and z.

We shall establish Result 1 (concretely, Theorem 4.8) using a proof by induction. The base case is straightforward and handled by the following lemma.

Lemma 4.2 (Base case)**.** *If a random tape adversarially robust algorithm* $\textsc{mif}(n,\ell)$ *uses at most* $\ell^2/(16n\ln 2)$ *bits of space and aborts with probability* $\leq \frac{1}{2}$*, then it makes a mistake with probability* $\geq \frac{1}{4}$*. Equivalently,*

$$\Delta(n,\ell,\gamma,z) \geq \frac{1}{4}\mathbb{1}_{z \leq \ell^2/(16n\ln 2)}\mathbb{1}_{\gamma \leq 1/2}\,. \tag{3}$$

Proof. Suppose that $\gamma \leq \frac{1}{2}$. Let $\mathcal{A} \in \textsc{Algs}(n,\ell,\gamma,z)$ be an algorithm with mistake probability $\delta \leq \frac{1}{4}$. Blurring the distinction between aborts and mistakes, and applying the space lower bound for random *oracle* algorithms (Theorem 3.5), we obtain

$$z \geq \frac{\ell^2}{4n\ln 2} + \log(1-\gamma-\delta) \geq \frac{\ell^2}{4n\ln 2} - 2 > \frac{\ell^2}{16n\ln 2}\,.$$

The latter inequality holds because $z \geq 1$ (trivially) and we have $\max(1, x-2) > x/4$. Taking the contrapositive, if $z \leq \ell^2/(16n\ln 2)$, then $\delta > 1/4$. This proves eq. (3). □

4.2 The Induction Step

The induction step consists of a reduction, using an adaptive adversary described in Listing 2, to prove a lower bound on the mistake probability. This is formalized in the next lemma and the rest of Section 4.2 is devoted to its proof.

Lemma 4.3 (Induction lemma)**.** *Let $1 \leq \ell < n$ and z be integers. Define, matching definitions in Listing 2,*

$$w := 2\left\lfloor 32\frac{zn}{\ell} \right\rfloor \qquad \text{and} \qquad t := \left\lfloor \frac{\ell}{64z} \right\rfloor . \tag{4}$$

If $z \geq 8$ and $t < w$, then:

$$\Delta\left(n, \ell, \frac{1}{2}, z\right) \geq \min\left(\frac{\ell}{2^7 nk}, \frac{1}{4}\Delta\left(w, t, \frac{1}{2}, z\right)\right) . \tag{5}$$

The Initial Random Prefix. The adversary begins by sending $\mathcal{A}$ a uniformly random sequence $X \in_R \text{SEQS}([n], q)$, i.e., a sorted sequence of q distinct elements of $[n]$; we'll eventually use $q = \lceil \ell/2 \rceil$. Let F be the *random function* where, for $x \in \text{SEQS}([n], q)$, $F(x)$ is the random state reached by $\mathcal{A}$ upon processing x, starting at its initial state; note that F is determined by the transition function T of $\mathcal{A}$ (see Section 1.1). Let Σ be the set of states of $\mathcal{A}$, so $|\Sigma| = 2^z$. For each $\sigma \in \Sigma$, define

$$H_\sigma := \left\{ i \in [n] : \Pr[i \in X \mid F(X) = \sigma] \leq \frac{q}{4n} \right\} , \tag{6}$$

which we can think of as the set of inputs that are "unlikely" to have been seen given that $\mathcal{A}$ has reached σ. A key part of our proof of Lemma 4.3 is the following lemma, which says that if Σ is small, then $\mathcal{A}$ doesn't have enough space to mark too many inputs as unlikely, so $H_{F(X)}$ is likely to be small. The lemma can be seen as a smoothed variant of the communication lower bound for AVOID.

Lemma 4.4. *Let Σ be a set with $|\Sigma| \leq 2^z$, let $1 \leq q \leq n$ be integers, let F be a random function that maps each sequence in $\text{SEQS}([n], q)$ to a random element of Σ, and let $X \in_R \text{SEQS}([n], q)$, chosen independently of F. For each $\sigma \in \Sigma$, define H_σ as in eq. (6). Then, for all $\alpha \in (0, 1)$,*

$$\Pr\left[\left|H_{F(X)}\right| \geq \hat{w}\right] \leq \alpha \qquad \text{where} \qquad \hat{w} := \left\lceil \frac{2\ln 2}{1 - \ln 2} \frac{z + 1 + \log\frac{1}{\alpha}}{q} n \right\rceil .$$

Proof. Consider a specific $\sigma \in \Sigma$. By linearity of expectation:

$$\mathbb{E}\left[\sum_{i \in H_\sigma} \mathbb{1}_{i \in X} \,\middle|\, F(X) = \sigma \right] = \sum_{i \in H_\sigma} \Pr[i \in X \mid F(X) = \sigma] \leq \frac{q}{4n} |H_\sigma| .$$

Then by Markov's inequality,

$$\Pr\left[\sum_{i \in H_\sigma} \mathbb{1}_{i \in X} \geq 2 \cdot \frac{q}{4n} |H_\sigma| \,\middle|\, F(X) = \sigma \right] \leq \frac{1}{2}$$

$$\text{which implies} \qquad \Pr\left[\sum_{i \in H_\sigma} \mathbb{1}_{i \in X} \leq \frac{q}{2n} |H_\sigma| \,\middle|\, F(X) = \sigma \right] \geq \frac{1}{2} .$$

Since X is drawn uniformly at random from $\textsc{seqs}([n],q)$, the random variables $\{\mathbb{1}_{i\in X}\}_{i\in[n]}$ are negatively associated, with $\mathbb{E}\,\mathbb{1}_{i\in X} = q/n$ for each $i \in [n]$. For any set $A \subseteq [n]$, we use the multiplicative Chernoff bound (Lemma 3.2) to bound the probability that X's overlap with A is much smaller than the expected value:

$$\Pr\left[\sum_{i\in A}\mathbb{1}_{i\in X} \le \left(1-\frac{1}{2}\right)\cdot\frac{q}{n}|A|\right] \le \left(\frac{e^{-1/2}}{(1/2)^{1/2}}\right)^{(q/n)|A|} = \exp\left(-\frac{1}{2}(1-\ln 2)\frac{q}{n}|A|\right).$$

We now bound

$$\Pr[F(X)=\sigma] \le \frac{\Pr\left[\sum_{i\in H_\sigma}\mathbb{1}_{i\in X} \le \frac{q}{2n}|H_\sigma|\right]}{\Pr\left[\sum_{i\in H_\sigma}\mathbb{1}_{i\in X} \le \frac{q}{2n}|H_\sigma| \;\middle|\; F(X)=\sigma\right]} \le 2\exp\left(-\frac{1}{2}(1-\ln 2)\frac{q}{n}|H_\sigma|\right).$$

Finally, let $B = \{\sigma\in\Sigma : |H_\sigma| \ge \hat{w}\}$. Then

$$\begin{aligned}\Pr\left[\left|H_{F(X)}\right| \ge \hat{w}\right] &= \sum_{\sigma\in B}\Pr[F(X)=\sigma]\\ &\le 2^z\cdot 2\exp\left(-\frac{1}{2}(1-\ln 2)\frac{q}{n}\hat{w}\right)\\ &\le 2^z\cdot 2\exp\left(-\left(z+1+\log\frac{1}{\alpha}\right)\ln 2\right) \le 2^z\cdot 2\cdot 2^{-\left(z+1+\log\frac{1}{\alpha}\right)} = \alpha.\end{aligned}$$

□

The Recursive Phases and Win-Win Argument. Let ρ denote the random state in Σ reached by $\mathcal{A}$ upon processing the random sequence X. Having observed $\mathcal{A}$'s outputs (transcript) in response to X, the adversary can compute $\mathcal{D}$, the distribution of ρ conditioned on this transcript. The adversary has $\lfloor \ell/2 \rfloor$ more items to send to $\mathcal{A}$ and it organizes these into phases of t items each (see eq. (4)). Recall, from the discussion in Section 2.2, that the adversary's goal is to identify a suitable sub-universe $W \subseteq [n]$ so that, in some phase, $\mathcal{A}$ can be seen as solving $\textsc{mif}(|W|,t)$ in this sub-universe. As it chooses a suitable input sequence for each phase, the adversary maintains the following objects to guide the choice:

- the evolving transcript of $\mathcal{A}$ given the inputs chosen so far;
- the corresponding distribution $\mathcal{D} \in \triangle[\Sigma]$;
- a set $Q \subseteq \Sigma$ where, for each $\sigma \in Q$, the set H_σ will be useful for determining W.

If the current set Q leads to a suitable W, the adversary's strategy for the next phase is to recursively run an optimal sub-adversary for $\textsc{mif}(|W|,t)$. If not, then (we shall show that) over the next phase, the adversary will be able to significantly shrink the set Q.

To make these ideas more concrete, we introduce some terminology below; these terms show up in Listing 2, which spells out the adversary's actions precisely.

Definition 4.5 (Divisive output sequence, Splitting adversary)**.** Let $\mathcal{A}$, Σ, and H_σ be as above and let $Q \subseteq \Sigma$. A sequence (of "outputs") $y \in [n]^t$ is said to be $\textsc{divisive}$ for Q if $|\{\sigma\in Q : y \subseteq H_\sigma\}| \le \frac{1}{2}|Q|$.

Say Υ is a t-length deterministic adversary, i.e., a function $\Upsilon\colon [n]^{\le t-1} \to [n]$. For each $\sigma\in\Sigma$, let $\textsc{Outs}(\sigma,\Upsilon)$ be the random variable in $[n]^t \cup \{\perp\}$ that gives the output if we run $\mathcal{A}$, starting at state σ, against the adversary Υ,[11] with $\perp$ indicating that $\mathcal{A}$ aborts. We say that Υ is α-$\textsc{splitting}$ for Q with respect to a distribution $\mathcal{D}\in\triangle[\Sigma]$ if

$$\Pr_{S\sim\mathcal{D}}[\textsc{Outs}(S,\Upsilon) \text{ is divisive for } Q] \ge \alpha,$$

with the convention that the value $\perp$ *is* divisive.[12]

[11] This means that if, after processing a few inputs, the algorithm has output sequence $v\in[n]^\star$, its next input will be $\Upsilon(v)$.

[12] The proof can also be made to work if one assumes $\perp$ is not divisive, but gives a weaker and more complicated result.

Notice that, knowing $\mathcal{D}$ and Q, the adversary can determine whether a given Υ is α-splitting. This is implicit in Line 8 of Listing 2.

Listing 2 An adversary for a random tape $\text{MIF}(n,\ell)$ algorithm.

ADVERSARY (against algorithm $\mathcal{A}$)

1: $w \leftarrow 2\lfloor 32zn/\ell \rfloor$; $h_{\max} \leftarrow 32z$; $t \leftarrow \lfloor \ell/(2h_{\max}) \rfloor$
2: $X \leftarrow$ a uniformly random sequence in $\text{SEQS}([n], \lceil \ell/2 \rceil)$.
3: **send** X to $\mathcal{A}$ and record the transcript of outputs
4: compute H_σ for each state σ using eq. (6), with $q = \lceil \ell/2 \rceil$
5: $Q_0 \leftarrow \{\sigma \in \Sigma : |H_\sigma| \le \frac{1}{2}w\}$
6: **for** h in $1, \ldots, h_{\max}$ **do**
7: $\quad \mathcal{D} \leftarrow$ distribution over Σ conditioned on the cumulative transcript so far
8: $\quad$ **if** $\exists$ a t-length deterministic adversary Υ that is $\frac{1}{2}$-splitting for Q_{h-1} w.r.t. $\mathcal{D}$ **then**
9: $\quad\quad$ **run** Υ against $\mathcal{A}$ and gather the transcript of outputs $y \in [n]^t$
10: $\quad\quad$ $Q_h \leftarrow \{\sigma \in Q_{h-1} : y \subseteq H_\sigma\}$ ▷ *have a $\ge \frac{1}{2}$ chance that $|Q_h| \le \frac{1}{2}|Q_{h-1}|$*
11: $\quad\quad$ **if** $Q_h = \varnothing$ **then fail**
12: $\quad$ **else**
13: $\quad\quad$ $W \leftarrow \{i \in [n] : |\{\sigma \in Q_{h-1} : i \in H_\sigma\}| \ge \frac{1}{2}|Q_{h-1}|\}$ ▷ *will show that $|W| \le w$*
14: $\quad\quad$ $W' \leftarrow W$ plus $w - |W|$ padding elements
15: $\quad\quad$ define algorithm $\mathcal{B}$ to behave like $\mathcal{A}$ conditioned on the transcript of inputs and outputs so far
16: $\quad\quad$ modify $\mathcal{B}$, changing every output outside W' to $\perp$ ▷ *thus $\mathcal{B}$ will abort in these cases*
17: $\quad\quad$ let Ξ be an adversary using inputs from W', maximizing the probability that $\mathcal{B}$ makes a mistake ▷ *can be computed using brute-force search*
18: $\quad\quad$ **run** adversary Ξ, sending t inputs in W'
19: $\quad\quad$ **return** ▷ *succeeded in causing $\mathcal{A}$ to have a high enough error probability*
20: **fail**

We proceed to prove the induction lemma.

Proof of Lemma 4.3. To prove the lower bound in eq. (5), we show that when the adversary in Listing 2 is run against a z-bit random-tape algorithm $\mathcal{A}$ for $\text{MIF}(n,\ell)$ which has $\le \frac{1}{2}$ worst-case probability of aborting, the probability that $\mathcal{A}$ makes a mistake is at least the right hand side of eq. (5). Note that the adversary feeds at most $\lceil \ell/2 \rceil + th_{\max} = \lceil \ell/2 \rceil + \lfloor \ell/(2h_{\max}) \rfloor h_{\max} \le \lceil \ell/2 \rceil + \lfloor \ell/2 \rfloor = \ell$ inputs to $\mathcal{A}$.

Consider a run of the adversary against $\mathcal{A}$. This is a random process, with some of the randomness coming from the adversary's choices (Line 2) and some coming from $\mathcal{A}$'s internal randomness. Let ρ be the state of $\mathcal{A}$ after X is sent. We now define a number of events, as follows.

- B_{UNSAFE} occurs if $\mathcal{A}$ produces an output in $[n] \setminus H_\rho$.
- B_{BIG} occurs if the state ρ has $|H_\rho| > \frac{1}{2}w$.
- B_{EMPTY} occurs if the adversary fails at Line 11.
- B_{TIMEOUT} occurs if the adversary fails at Line 20.
- B_{ABORT} occurs if $\mathcal{A}$ aborts *before* the adversary reaches Line 18.
- R_{ERROR} occurs if $\mathcal{A}$ makes a mistake *while* the adversary is executing Line 18.

We will consider each of the events listed above.

(Event B_{UNSAFE}) If B_{UNSAFE} occurs, then some $i \in [n] \setminus H_\rho$ is output and, in view of eq. (6), the set X from Line 2 has the property that

$$\Pr[i \in X \mid \mathcal{A} \text{ reaches } \rho] \geq \frac{\lceil \ell/2 \rceil}{4n} \geq \frac{\ell}{8n}.$$

Consequently, the probability that $\mathcal{A}$ makes a mistake by producing an output from X (which would be a non-missing item) is $\geq \Pr[B_{\text{UNSAFE}}] \cdot \ell/(8n)$.

If $\Pr[B_{\text{UNSAFE}}] > 1/(16)$, we then have $\Delta(n, \ell, \frac{1}{2}, z) \geq \ell/(2^7 n)$, which implies eq. (5) leaving nothing more to prove. Therefore, for the rest of this proof we will consider the case in which

$$\Pr[B_{\text{UNSAFE}}] \leq \frac{1}{16}. \tag{7}$$

(Event B_{BIG}) We apply Lemma 4.4 with $q = \lceil \ell/2 \rceil$, $\alpha = 1/16$, F being the random function that maps each $x \in \text{SEQS}([n], q)$ to the random state reached by $\mathcal{A}$ upon processing x, starting at its initial state, and

$$\begin{aligned}
\hat{w} = \left\lceil \frac{2\ln 2}{1-\ln 2} \frac{z+1+\log 16}{\lceil \ell/2 \rceil} n \right\rceil &\leq 1 + \left\lfloor \frac{2\ln 2}{1-\ln 2} \frac{z+1+\log 16}{\lceil \ell/2 \rceil} n \right\rfloor \\
&\leq 1 + \left\lfloor \frac{8\ln 2}{1-\ln 2} \frac{z+1+\log 16}{\ell} n \right\rfloor \\
&\leq 1 + \left\lfloor 19 \frac{z+5}{\ell} n \right\rfloor \leq 1 + \left\lfloor 32 \frac{zn}{\ell} \right\rfloor = \frac{1}{2} w + 1,
\end{aligned}$$

since $z \geq 8$. As $\frac{1}{2}w$ is an integer,

$$\Pr[B_{\text{BIG}}] = \Pr\left[|H_\rho| > \frac{1}{2}w\right] = \Pr\left[|H_\rho| \geq \frac{1}{2}w + 1\right] \leq \Pr\left[|H_\rho| \geq \hat{w}\right] \leq \frac{1}{16}. \tag{8}$$

(Event B_{ABORT}) There are exactly exactly three spots in Listing 2 where the algorithm can abort: Lines 3, 9 and 18, when the adversary is feeding it inputs. By assumption, $\mathcal{A}$'s worst case probability of aborting, against *any* adversary, is at most $\frac{1}{2}$. Therefore, in particular,

$$\Pr[B_{\text{ABORT}}] \leq \frac{1}{2}. \tag{9}$$

When the algorithm does abort, we stop running the adversary, so B_{ABORT}, B_{EMPTY}, and B_{TIMEOUT} are mutually exclusive.

(Event B_{EMPTY}) For this to happen, at some point we must have $Q_h = \varnothing$. At that point, the state ρ must not be in Q_h, in which case either $\rho \notin Q_0$ or ρ was filtered out of Q_h on Line 10. By the definition of Q_0, $\rho \notin Q_0$ iff B_{BIG} holds. On the other hand, filtering ρ out of Q_h requires that the algorithm produce an output outside H_ρ, which can only happen if either B_{UNSAFE} or B_{ABORT} occurs. Since B_{ABORT} is mutually exclusive with B_{EMPTY}, we conclude that

$$B_{\text{EMPTY}} \Rightarrow B_{\text{UNSAFE}} \vee B_{\text{BIG}}. \tag{10}$$

(Event B_{TIMEOUT}) We bound the probability that the adversary will fail using Line 20. For this to happen, the adversary must have picked $h_{\max}$ splitting adversaries, but fewer than $z+1$ of them must have produced a divisive output. (If there is a divisive output in round h, then $|Q_h| \leq \frac{1}{2}|Q_{h-1}|$; if not, then

$|Q_h| \le |Q_{h-1}|$. Thus with $z+1$ divisive outputs, $|Q_{h_{\max}}| \le |Q_0|/2^{z+1} \le |\Sigma|/2^{z+1} \le \frac{1}{2} < 1$, in which case the adversary would have failed at Line 11 instead.)

For each $h \in [h_{\max}]$, let X_h be the $\{0,1\}$ indicator random variable for the event that a divisive output is found in the hth step. (If the hth step did not occur or no splitting adversary was found, set $X_h = 1$.[13]) Since in the hth step, a splitting adversary for the distribution for the current state of the algorithm, conditioned on the transcript so far, is chosen, then $\mathbb{E}[X_h \mid X_1, \ldots, X_{h-1}] \ge 1/2$. Applying Lemma 3.1 gives:

$$\begin{aligned}
\Pr\left[\sum_{h\in[h_{\max}]} X_h < z+1\right] &= \Pr\left[\sum_{h\in[h_{\max}]} X_h \le \left(1-\left(1-\frac{2z}{h_{\max}}\right)\right)\frac{h_{\max}}{2}\right] \\
&\le \exp\left(-\frac{1}{2}\left(1-\frac{2z}{h_{\max}}\right)^2 \frac{h_{\max}}{2}\right) \\
&\le \exp\left(-\frac{1}{8}\frac{h_{\max}}{2}\right) && \text{since } h_{\max} = 32z \ge 4z \\
&\le \frac{1}{8}. && \text{since } h_{\max} \ge 16\ln 8
\end{aligned}$$

Thus $\Pr[B_{\text{TIMEOUT}}] \le 1/8$.

Combining this with eqs. (7) to (10) gives:

$$\begin{aligned}
\Pr[B_{\text{EMPTY}} \vee B_{\text{TIMEOUT}} \vee B_{\text{ABORT}}] &\le \Pr[B_{\text{UNSAFE}} \vee B_{\text{BIG}} \vee B_{\text{TIMEOUT}} \vee B_{\text{ABORT}}] \\
&\le \Pr[B_{\text{UNSAFE}}] + \Pr[B_{\text{BIG}}] + \Pr[B_{\text{TIMEOUT}}] + \Pr[B_{\text{ABORT}}] \\
&\le \frac{1}{16} + \frac{1}{16} + \frac{1}{8} + \frac{1}{2} = \frac{3}{4}. \qquad (11)
\end{aligned}$$

(Event R_{ERROR}) Let E be the event that the adversary executes Line 18. Notice that R_{ERROR} can only occur when E occurs. Note also that

$$E = \neg B_{\text{EMPTY}} \wedge \neg B_{\text{TIMEOUT}} \wedge \neg B_{\text{ABORT}},$$

so eq. (11) implies that

$$\Pr[E] \ge \frac{1}{4}. \qquad (12)$$

Suppose that E does occur. We now make two claims: (a) that $|W| \le w$, and (b) that $\mathcal{B} \in \text{ALGS}(w, t, \frac{1}{2}, z)$. For the first claim, note that by Line 5, the set Q_0 only contains states $\sigma \in \Sigma$ with $|H_\sigma| \le \frac{1}{2}w$. The same bound on $|H_\sigma|$ holds for all $\sigma \in Q_{h-1}$ because $Q_{h-1} \subseteq Q_0$, thanks to Line 10. By the definition of W (Line 13),

$$\begin{aligned}
|W| &= \left|\left\{i \in [n] : \frac{|\{\sigma \in Q_{h-1} : i \in H_\sigma\}|}{|Q_{h-1}|} \ge \frac{1}{2}\right\}\right| \\
&\le \sum_{i\in[n]} \frac{2|\{\sigma \in Q_{h-1} : i \in H_\sigma\}|}{|Q_{h-1}|} \\
&= \frac{2}{|Q_{h-1}|} \sum_{\sigma\in Q_{h-1}} |H_\sigma| \le \frac{2}{|Q_{h-1}|} \cdot |Q_{h-1}| \cdot \frac{1}{2}w = w,
\end{aligned}$$

[13] Note that this definition accounts for the cases where the algorithm aborts: if it aborts on h, by Definition 4.5 this is interpreted as divisive, and the following steps do not occur, so $X_h = X_{h+1} = \ldots, X_{h_{\max}} = 1$. Then the event $\{\sum X_h < z+1\}$ slightly overestimates the probability of B_{TIMEOUT}; it would be more accurate to have $\mathcal{A}$ aborting produce $X_h = \infty$.

which proves claim (a).

For the second claim, consider the sub-algorithm $\mathcal{B}'$ defined as $\mathcal{B}$ just before the modification at Line 16. Then, for a particular adversary Υ, running Υ against $\mathcal{B}$ causes an abort exactly when running Υ against $\mathcal{B}'$ causes either an abort or an output outside W'. In the iteration of the **for** loop that caused E, since we reached the **else** branch, there was no deterministic splitting adversary with respect to $\mathcal{D}$. Thus for *any* deterministic adversary Υ, by Definition 4.5,

$$\Pr_{\hat{\sigma}\sim\mathcal{D}}[\textsc{Outs}(\hat{\sigma},\Upsilon) \text{ is divisive for } Q_{h-1}] \le \frac{1}{2}.$$

Let y be a realization of the random variable $\textsc{Outs}(\hat{\sigma},\Upsilon)$; note that $y \in [n]^t \cup \{\perp\}$. If a given y is not divisive, then $y \in [n]^t$ and, for each $i \in y$,

$$|\{\sigma \in Q_{h-1} : i \in H_\sigma\}| \ge |\{\sigma \in Q_{h-1} : y \subseteq H_\sigma\}| \ge \frac{1}{2}|Q_{h-1}|,$$

which implies that $i \in W$. Thus in fact $y \subseteq W$. It follows that the probability that running Υ against $\mathcal{B}'$ causes an abort or an output outside W (which is a subset of W') is at most $\frac{1}{2}$. Viewing $\mathcal{B}$ as a random-tape MIF algorithm handling input streams of length at most t, with items from W', in the terminology of Definition 4.1, we have $\mathcal{B} \in \textsc{Algs}(w,t,\frac{1}{2},z)$. This proves claim (b).

Since Ξ is picked to maximize the probability of $\mathcal{B}$ making a mistake, we have $\Pr[R_{\text{ERROR}} \mid E] \ge \Delta(w,t,\frac{1}{2},z)$. Using eq. (12), we obtain

$$\Pr[R_{\text{ERROR}}] \ge \Pr[E]\,\Delta\left(w,t,\frac{1}{2},z\right) \ge \frac{1}{4}\Delta\left(w,t,\frac{1}{2},z\right).$$

Combining this lower bound with the lower bound for the case where $\Pr[B_{\text{UNSAFE}}] > \frac{1}{16}$ (considered just before eq. (7)), we obtain

$$\Delta\left(n,\ell,\frac{1}{2},z\right) \ge \min\left(\frac{\ell}{2^7 n}, \frac{1}{4}\Delta\left(w,t,\frac{1}{2},z\right)\right). \qquad \square$$

4.3 Calculating the Lower Bound

Lemma 4.6. *Let $1 \le \ell < n$. For any integer $k \ge 1$, say that z is an integer satisfying $z \le \frac{1}{256}\ell^{1/k}$. Then:*

$$\Delta(n,\ell,0,z) > \min\left(\frac{\ell}{2^7 n}, \frac{1}{4^k}\mathbb{1}_{z\le L}\right) \qquad \textit{where} \qquad L = \frac{1}{64}\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}. \tag{13}$$

Consequently, algorithms for MIF with $\le \min(\frac{\ell}{2^7 n}, 4^{-k})$ error require $> L$ bits of space.

Proof of Lemma 4.6. Let $n_1 = n$ and $\ell_1 = \ell$, and for $i = 2,\dots,k$, set $n_i = 2\left\lfloor 32\frac{z n_{i-1}}{\ell_{i-1}}\right\rfloor$ and $\ell_i = \left\lfloor \frac{\ell_{i-1}}{64z}\right\rfloor$. This matches the definitions used in Lemma 4.3. As we have been promised that $z \le \frac{1}{256}\ell^{1/k}$, we have in particular that:

$$(256z)^k \le \ell \qquad \text{which implies} \qquad \ell_1 \ge \cdots \ge \ell_k \ge 256z.$$

By Lemma 3.6, we only need to consider the case $z \ge \log(\ell+1)$, as otherwise $\Delta(n,\ell,0,z) = 1$. Thus, if $k \ge 2$, we have:

$$z \ge \log(\ell+1) \ge k\log(256z) \ge k\log(256) = 16 \ge 8.$$

(If $k = 1$, then eq. (13) follows immediately from Lemma 4.2.)

Since $\Delta(n,\ell,0,z) \geq \Delta(n,\ell,1/2,z)$, it suffices to lower bound the case where algorithms are permitted up to $1/2$ abort probability. We will lower bound $\Delta(n,\ell,1/2,z)$ by recursively applying Lemma 4.3 $k-1$ times, and then applying Lemma 4.2. This yields:

$$\begin{aligned}
\Delta(n,\ell,1/2,z) &\geq \min\left(\frac{\ell_1}{2^7 n_1}, \frac{1}{4}\Delta(n_1,\ell_1,1/2,z)\right) \\
&\geq \min\left(\frac{\ell_1}{2^7 n_1}, \frac{1}{4}\min\left(\frac{\ell_2}{2^7 n_2}, \dots \frac{1}{4}\min\left(\frac{\ell_{k-1}}{2^7 n_{k-1}}, \frac{1}{4}\mathbb{1}_{z\leq \ell_k^2/(16 n_k \ln 2)}\right)\cdots\right)\right) \\
&\geq \min\left(\frac{\ell_1}{2^7 n_1}, \frac{1}{4}\frac{\ell_2}{2^7 n_2}, \dots, \frac{1}{4^{k-1}}\frac{\ell_{k-1}}{2^7 n_{k-1}}, \frac{1}{4^k}\mathbb{1}_{z\leq \ell_k^2/(16 n_k \ln 2)}\right).
\end{aligned}$$

Only the first and last terms of the minimum are significant, because the terms for $i = 2,\dots,k-1$ are all dominated by the first term:

$$\frac{\ell_i}{n_i} = \frac{\left\lfloor \frac{\ell_{i-1}}{64z}\right\rfloor}{2\left\lfloor 32\frac{z n_{i-1}}{\ell_{i-1}}\right\rfloor} \geq \frac{1}{2}\frac{\ell_{i-1}}{64z}\frac{\ell_{i-1}}{64 z n_{i-1}} \geq \frac{\ell_{i-1}}{2(64z)^2}\frac{\ell_{i-1}}{n_{i-1}} \geq 4\frac{\ell_{i-1}}{n_{i-1}},$$

where in the last step, we used the fact that $\ell_{i-1} \geq \ell_{k-2} \geq (256z)^2$. Thus:

$$\Delta(n,\ell,0,z) \geq \min\left(\frac{\ell}{2^7 nk}, \frac{1}{4^k}\mathbb{1}_{z\leq \ell_k^2/(16 n_k \ln 2)}\right).$$

We have almost proven eq. (13). It remains to lower bound $\mathbb{1}_{z\leq \ell_k^2/(16 n_k \ln 2)}$ by $\mathbb{1}_{z\leq L}$ for some L. We do so by proving $z > \ell_k^2/(16 n_k \ln 2)$ implies $z > L$. As a consequence of the definitions, we have:

$$\ell_i = \left\lfloor \frac{\ell_{i-1}}{64z}\right\rfloor = \left\lfloor \frac{\left\lfloor \frac{\ell_{i-2}}{64z}\right\rfloor}{64z}\right\rfloor = \cdots = \left\lfloor \frac{\ell}{(64z)^{i-1}}\right\rfloor \qquad \text{and} \qquad n_i \leq 64\frac{z n_{i-1}}{\ell_{i-1}},$$

and thus:

$$\begin{aligned}
z &> \frac{\ell_k^2}{16\ln 2}\frac{1}{n_k} \geq \frac{\ell_k^2}{16}\frac{\prod_{k=1}^{k-1}\ell_k}{(64z)^{k-1}n} \\
&\geq \frac{\ell^{k+1}}{16\cdot(64z)^{2(k-1)+(k-2)+(k-3)+\cdots+1+0}(64z)^{k-1}n} \\
&= \frac{\ell^{k+1}}{16\cdot(64z)^{(k^2+3k-4)/2}n}.
\end{aligned}$$

Rearranging to put all the z terms on the left gives:

$$64z\cdot(64z)^{\frac{k^2+3k-4}{2}} \geq 64\frac{\ell^{k+1}}{16n},$$

which implies

$$z > \frac{1}{64}\left(\frac{64\ell^{k+1}}{16n}\right)^{\frac{2}{k^2+3k-2}} \geq \frac{1}{64}\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}. \qquad \square$$

Now, say a random tape algorithm has error probability $\leq \frac{\ell}{2^7 n}$ against any adaptive adversary, and uses z bits of space. If $\ell \geq n^{2/3}$, it is easy to show that the optimal value of k with which to apply Lemma 4.6 is 1. Otherwise, for all k, we have two cases: if $z \leq \frac{1}{256}\ell^{1/k}$, then we can apply Lemma 4.6. By Lemma 3.6, $z\log(\ell+1) \geq 1$, which implies $(256)^k \leq \ell$ and thus $k \leq \left\lfloor \frac{1}{8}\log \ell \right\rfloor$. As the algorithm has

$$\text{error} \leq \frac{\ell}{2^7 n} \leq \min\left(\frac{\ell}{2^7 n}, \frac{1}{2^7 \ell^{1/2}}\right) \leq \min\left(\frac{\ell}{2^7 n}, \frac{1}{4^k}\right),$$

by Lemma 4.6 the algorithm's space usage must satisfy:

$$z \geq \frac{1}{64}\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}.$$

Since we either have this lower bound on z, or $z > \frac{1}{256}\ell^{1/k}$, it follows that for any integer $k \geq 1$:

$$z \geq \frac{1}{256}\min\left(\ell^{1/k}, \left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}\right).$$

We can take the maximum over all values of k to obtain:

$$z \geq \frac{1}{256}\max_{k\in\mathbb{N}}\min\left(\ell^{1/k}, \left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}\right).$$

The right hand side can be simplified with the following lemma, whose proof is mostly calculation and is deferred to Appendix A.2:

Lemma 4.7.

$$\max_{k\in\mathbb{N}}\min\left(\ell^{1/k}, \left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}\right) \geq \max_{k\in\mathbb{N}}\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}} \geq \ell^{\frac{15\log\ell}{32\log n}} \tag{14}$$

Summarizing, we obtain the following theorem.

Theorem 4.8. *Random tape δ-error adversarially robust algorithms for* $\text{MIF}(n,\ell)$ *require*

$$\Omega\left(\max_{k\in\mathbb{N}}\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}\right) = \Omega\left(\ell^{\frac{15}{32}\log_n \ell}\right)$$

bits of space, for $\delta \leq \ell/(2^7 n)$. □

4.4 Remarks on the Lower Bound

To prove the above lower bound, we required $\delta \leq \frac{\ell}{2^7 n}$. For larger values of δ, random tape algorithms can be much more efficient. For example, there is a $(\log t)$-space algorithm for $\text{MIF}(n,\ell)$ with $O(\ell^2/t)$ error probability, which on every input randomly picks a new state (and output value) from $[t]$.

That being said, if a random tape algorithm $\mathcal{A}$ with error $\leq \delta$ provides the additional guarantee that it never makes a mistake (i.e, either produces a correct output or aborts),[14] one can construct a new algorithm

[14]This guarantee essentially rules out the possibility of algorithms that randomly and blindly guess outputs. Most of the algorithms for MIF in this paper and in [Sto23] provide this "zero-mistake" guarantee.

$\mathcal{B}$ with error $O(\frac{1}{n})$ by running $\Theta(\frac{\log n}{\log 1/\delta})$ parallel copies of $\mathcal{A}$ and reporting outputs from any copy that has not yet aborted. Proving a space lower bound for $\mathcal{B}$ then implies a slightly weaker one for $\mathcal{A}$.

The lower bound of Theorem 4.8 is not particularly tight, and we suspect it can be improved to match the upper bound within $\text{polylog}(\ell, n)$ factors. There are at least two scenarios that we suspect algorithms must behave similarly to, in which we could do better than the current adversary's reduction to $\text{MIF}(w, \Theta(\ell/z))$

- If a constant fraction of the next $\ell/2$ outputs are contained in W, we could (essentially) reduce to $\text{MIF}(w, \ell/2)$.
- If on each search step of length $\Theta(\ell/z)$, the outputs of the algorithm are concentrated in a new set of size $\Theta(w/z)$, then we could (essentially) reduce to $\text{MIF}(\Theta(w/z), \Theta(\ell/z))$.

The adversary of Listing 2 runs in doubly exponential time, and requires knowledge of the algorithm. The former condition cannot be improved by too much: if one-way functions exist, one could implement the random oracle algorithm for $\text{MIF}(n, \ell)$ from [Sto23] using a pseudo-random generator that fools all polynomial-time adversaries. One can also prove by minimax theorem that universal adversaries for (random tape or otherwise) $\text{MIF}(n, \ell)$ algorithms can not be used to prove any stronger lower bounds than the one for random oracle algorithms.

5 The Random Tape Upper Bound

In this section, we describe an adversarially robust random tape algorithm for $\text{MIF}(n, \ell)$ which obtains error $\leq \delta$. See Section 2.1 for a high level overview. The algorithm, shown in Listing 3, can be implemented for almost all pairs $\ell < n$, requiring only $\ell \leq n/64$ and $\ell \geq 4$ for its parameters to be meaningful. It can be seen as a multi-level generalization of the random oracle algorithm from [Sto23].

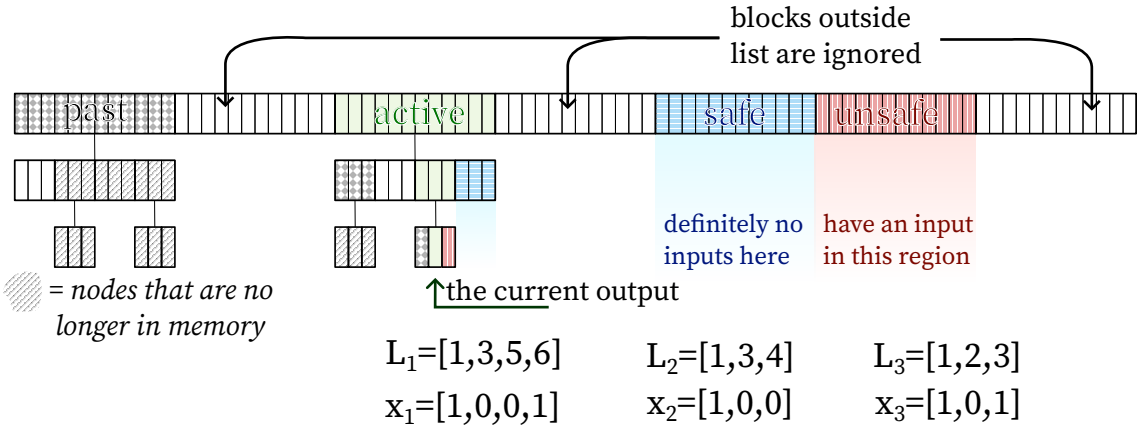

Figure 3: Diagram showing the state of the algorithm in Listing 3 and how it relates to the parts of the implicit random tree that the algorithm traverses. Positions on the horizontal axis correspond to different integers in $[n]$. To keep the example legible, we set parameters $d = 3$, $w_1 = 7, w_2 = 4, w_3 = 3$, and $b_1 = 4, b_2 = 3, b_3 = 3$.

In order to prove that the algorithm in Listing 3 is correct, we will need some additional notation. Let $d, \alpha, b_1, \ldots, b_d, w_1, \ldots, w_d$ be as defined in Listing 3. It is helpful to view this algorithm as traversing over the leaves of a random tree of height d, in which:

- Every node v in the tree is associated with a subset S_v of $[n]$. We say a node is at level i if it is at depth $i-1$. All nodes at a given level have disjoint associated subsets.

- The "root" ρ of the tree has S_ρ of size $\prod_{i=1}^{d} w_i$, and is at level 1
- Each node v at level i (depth $i-1$) has b_{i+1} children; the set S_v is partitioned into w_{i+1} parts of equal size, and each child of v is associated with a random and unique one of these parts
- Each leaf node u, at depth d, is associated with a set of size 1, i.e., a single and unique integer in $[n]$. There are $\prod_{i=1}^{d} b_i$ leaf nodes in total.

See for example Figure 3. The algorithm maintains a view of just the branch of the tree from the root to the current leaf node. Its output will be the number associated to this leaf. For each node v on this branch, at level $i \in [d]$ (depth $i-1$), it keeps a record of the positions $L_i \in [w_i]^{b_i}$ of its children, and a record $x_i \in \{0,1\}^{b_i}$ indicating their status. There are four categories for child nodes:

- A node is PAST if the traversal over the tree passed through and leaf the node; past nodes are marked with a 1 in x_i.
- A node is ACTIVE if it is on the branch to the current leaf node; this is the node with the lowest index which is marked with a 0 in x_i.
- A node v is SAFE if it comes after the active node, and the adversary has never sent an input in S_v; safe nodes are marked with a 0 in x_i.
- A node v is UNSAFE if it comes after the active node, and the adversary did send an input in S_v; unsafe nodes are marked with a 1 in x_i.

The algorithm maintains these records as the adversary sends new inputs, marking safe child nodes v as unsafe if an element in S_v is received. The current leaf node is found by, from the root, following the chain of active nodes. If the adversary sends the value of the current leaf node, the algorithm will mark it by setting the corresponding entry in x_d to 1, thereby changing the value of the current active node. If every element of x_d is a now 1, this means that the adversary has sent an input for every child of the level d node on the current branch, so the algorithm marks the current active child of the level $d-1$ node with a 1, thereby moving the current branch to use a new level d node, u, which is *safe*.[15] It the "loads the positions of the children of u"—the tree being randomly generated, this is implemented by L_d being randomly sampled and x_d being reset to be all zeros—and proceeds.

While there are $\prod_{i=1}^{d} b_i$ leaf nodes in the ideal random tree, the algorithm's traversal of them may skip a fraction, because they (or one of their ancestors) were marked as unsafe. We say that such leaf nodes, along with those which were once active, have been KILLED.

Listing 3 uses the following lemma to set some of its parameters; the specific rounding scheme for the values $b_2, \ldots, b_{d-1}$ ensures that b_1 can decrease relatively smoothly as ℓ decreases. (Setting all $b_2 = \ldots = b_{d-1}$ to $\lfloor \alpha \rfloor$ can lead to having b_1 be significantly larger than necessary (by up to a factor $(3/2)^d = \ell^{O(1)}$); setting all $b_2 = \ldots = b_{d-1}$ to $\lceil \alpha \rceil$ would violate the $\prod_{i=1}^{d} w_i \le n$ constraint.)

Lemma 5.1. *Let $\alpha \ge 1$. Then for all $k \ge 0$, there exists an integer u depending on α and k so that*

$$\alpha^k \le \lceil \alpha \rceil^u \lfloor \alpha \rfloor^{k-u} \le 2\alpha^k .$$

Proof of Lemma 5.1. If α is an integer, we are done. Otherwise, with

$$u = \left\lceil \frac{k \log(\alpha/\lfloor \alpha \rfloor)}{\log(\lceil \alpha \rceil/\lfloor \alpha \rfloor)} \right\rceil , \quad \text{we have} \quad \lceil \alpha \rceil^u \lfloor \alpha \rfloor^{k-u} = \lfloor \alpha \rfloor^k (\lceil \alpha \rceil/\lfloor \alpha \rfloor)^u \ge \lfloor \alpha \rfloor^k (\lceil \alpha/\lfloor \alpha \rfloor \rceil)^k = \alpha^k .$$

Similarly, $\lceil \alpha \rceil^u \lfloor \alpha \rfloor^{k-u} \le \alpha^k \lceil \alpha \rceil/\lfloor \alpha \rfloor$, which is $\le 2\alpha^k$ because $\alpha \ge 1$ implies $\lceil \alpha \rceil/\lfloor \alpha \rfloor \le 2$. □

[15] If the level $d-1$ node has no children marked with a 0 after this, we repeat the process at level $d-2$, and so on.

Listing 3 Adversarially robust random tape algorithm for $\mathrm{MIF}(n,\ell)$ with error $\leq \delta$

Requirements: $\ell \leq n/64$ and $\ell \geq 4$.

Parameters: $d = \min(\lceil \log \ell \rceil, \left\lfloor 2\frac{\log(n/4)}{\log 16\ell} \right\rfloor)$.

$$\alpha = \begin{cases} 2 & \text{if } \lceil \log \ell \rceil < \left\lfloor 2\frac{\log(n/4)}{\log 16\ell} \right\rfloor \\ \frac{(4\ell)^{2/(d-1)}}{(n/4)^{2/(d(d-1))}} & \text{otherwise} \end{cases}$$

Let u be chosen via Lemma 5.1, so that $\alpha^{d-2} \leq \prod_{i=2}^{d-1} b_i \leq 2\alpha^{d-2}$

$b_2 = \ldots = b_u = \lceil \alpha \rceil$, and $b_{u+1} = \ldots = b_{d-1} = \lfloor \alpha \rfloor$

$b_1 = \min(\ell+1, \lceil 8\alpha \rceil + \lceil 3\log 1/\delta \rceil)$; and $b_d = \left\lceil \frac{\ell}{\alpha^{d-1}} \right\rceil$

$w_1 = 16\ell$; and for each $i \in \{2,\ldots,d\}$, $w_i = \prod_{j=i}^{d} b_j$.

Let $\iota : [w_1] \times [w_2] \times \cdots \times [w_d] \to [n]$ be an arbitrary injective function

Initialization:

for $i \in [d]$ **do**
    $L_i \leftarrow$ random sequence without repetition in $[w_i]^{b_i}$
    $x_i \leftarrow (0,\ldots,0) \in \{0,1\}^{b_i}$

Update($a \in [n]$):

if $a \notin \iota^{-1}[n]$ **then return** ▷ *Any integer not in $\iota^{-1}[n]$ can never be an output*
$v_1,\ldots,v_d = \iota^{-1}(a)$ ▷ *Map input into $[w_1] \times \ldots \times [w_d]$*
For $i \in [d]$, define $c_i = \min|\{j : x_i[j] = 0\}|$
if for all $i \in [d]$, $v_i = L_i[c_i]$ **then**
    ▷ *Move to the next leaf node, sampling new child node positions as necessary* ◁
    for $i = d,\ldots,1$ **do**
        $x_i[c_i] \leftarrow 1$
        if x_i is the all-1s vector **then**
            if $i = 1$ **then abort** ▷ *If we reach $i = 1$, then even the root node is full*
            $L_i \leftarrow$ random sequence without repetition in $[w_i]^{b_i}$
            $x_i \leftarrow (0,\ldots,0)$
        else break
else
    ▷ *Mark a branch as unsafe, if there was a hit* ◁
    Let j be the smallest integer in $[d]$ for which $v_j \neq L_j[c_j]$.
    if $\exists y \in [b_j]$ for which $L_j[y] = v_j$ **then**
        $x_j[y] \leftarrow 1$

Output $\to [n]$:

For $i \in [d]$, define $c_i = \min|\{j : x_i[j] = 0\}|$
return $\iota[(L_1[c_1], L_2[c_2], \ldots, L_d[c_d])]$

The following lemma is straightforward but tedious, and we defer its proof to Appendix A.3.

Lemma 5.2. *The parameters of Listing 3 satisfy the following conditions:*

$$\prod_{i=2}^{d} b_i \geq \frac{\ell}{\alpha}, \tag{15}$$

$$\prod_{i=2}^{d} b_i \leq \frac{4\ell}{\alpha}, \tag{16}$$

$$\prod_{i\in[d]} w_i \leq n. \tag{17}$$

We now prove the main lemma:

Lemma 5.3. *Listing 3 has error $\leq \delta$ in the adversarial setting.*

Proof of Lemma 5.3. We prove, using a charging scheme, that the probability of all leaf nodes in the random tree traversed by the algorithm being killed is $\leq \delta$.

The input of the adversary at any step falls into one of $d+2$ categories. For each $i \in [d]$, it could add an input which intersects the list of unrevealed child positions of the level i node, possibly killing $\prod_{j=i+1}^{d} b_i$ leaf nodes if it guesses correctly. It could also send the value of the current leaf node, thereby killing it (and only it). Finally, the adversary's input could be entirely wasted (outside $\iota([w_1] \times \ldots \times [w_d]$, repeating an input it made before, or in the region corresponding to one of the past nodes in the random tree); then no leaf nodes would be killed.

As the algorithm proceeds, for each node in the random tree (other than the root), we accumulate charge. When the algorithm's current branch changes to use new nodes, the charge on the old nodes is kept, and the new nodes start at charge 0.

When the adversary makes an input that is handled by level i ("query" at level i), for $i \in \{2,\ldots,d\}$, it *first* deposits one unit of charge at the active level i node. Then, if the query was a hit (i.e, ruled out some future subtree and made a child of a node in the current branch change from "safe" to "unsafe"), increase the number of killed nodes by the number of leaves for the subtree (namely, $\prod_{j=i+1}^{d} b_j$), and remove up to that amount of charge from the node. The definitions of $(w_j)_{j=2,\ldots,d}$ ensure that $\prod_{j=i+1}^{d} b_j = w_j/b_j$.

For the tth query, let K_t be the number of killed leaf nodes on the query, minus any accumulated charge on the node. Say the adversary picks a node at level i for $i \in \{2,\ldots,d-1\}$, and that node has $\hat{w}$ unexplored subtree regions (i.e, neither revealed because the algorithm produced outputs in them, nor because there was there a query at that subtree region in the past), and $\hat{b}$ gives the number of subtrees within this unexplored region. If $\hat{b} = 0$, $\mathbb{E}[K_t | K_1,\ldots,K_{t-1}] = 0$. Otherwise, let u be the number of subtrees which were revealed by the algorithm so far; we have $\hat{w} \leq w_j - u$ and $\hat{b} \leq b_j - u$. Then when we condition on the past increases in charge, the subtree regions within the unexplored region are still uniformly random; hence the probability of hitting a subtree is $\hat{b}/\hat{w}$. The number of leaf nodes killed by a hit is w_j/b_j. The total charge currently at the node must be $\geq (w_j - u - \hat{w}) - (\frac{w_j}{b_j})(b_j - u - \hat{b}) = \hat{b}\frac{w_j}{b_j} - \hat{w} + (\frac{w_j}{b_j} - 1)u \geq \hat{b}\frac{w_j}{b_j} - \hat{w}$, since each removed node consumes at most $\frac{w_j}{b_j}$ of the existing charge. Consequently, the increase in killed leaf nodes if we hit is $\max(0, \frac{w_j}{b_j} - \max(0, \hat{b}\frac{w_j}{b_j} - \hat{w}))$, so the expected payoff is[16]:

$$\mathbb{E}[K_t \mid K_1,\ldots,K_{t-1}] = \frac{\hat{b}}{\hat{w}} \max\left(0, \frac{w_j}{b_j} - \max\left(0, \hat{b}\frac{w_j}{b_j} - \hat{w}\right)\right) \underset{\text{by Lemma 5.4}}{\leq} 1.$$

[16] When the level is d and $b_j = w_j$, we in fact we have $K_t = 1$ always; but we do not need this stronger fact.

If the level is 1, then let $J \subseteq [16\ell]$ give the set of probed subtree positions, and H give the set of revealed subtree positions; since there are $\le \ell$ queries, $|J|, |H|$ are both $\le \ell$, and the probability of a query in an unexplored region to hit is $\le \frac{b_1}{16\ell - |J \cup H|} \le \frac{b_1}{14\ell}$. The charging scheme does not apply, so

$$\mathbb{E}[K_t \mid K_1, \ldots, K_{t-1}] \le \frac{b_1}{14\ell} \cdot \prod_{j=2}^{d} b_j .$$

Thus in all cases, $\mathbb{E}[K_t \mid K_1, \ldots, K_{t-1}] \le \max(1, \prod_{j=1}^{d} b_j / 14\ell)$.

The total charge deposited on mid-level nodes is $\le \ell$. The algorithm is guaranteed to succeed if the total number of leaves killed is less than the total number of leaves; i.e, if $\sum_{i \in [\ell]} K_t + \ell \le \prod_{j=1}^{d} b_j$. Note that by Lemma 5.2 and the definition of b_1, $\prod_{j=1}^{d} b_j \ge b_1 \ell / \alpha \ge 8\ell$. Consequently,

$$\ell + 7\mathbb{E}\left[\sum_{t=1}^{\ell} K_t\right] \le 8 \max\left(\ell, \frac{1}{14}\prod_{j=1}^{d} b_j\right) \le 8 \max\left(\frac{1}{8}, \frac{1}{14}\right) \prod_{j=1}^{d} b_j \le \prod_{j=1}^{d} b_j .$$

Now let $D_t = K_t / \prod_{j=2}^{d} b_j$, so that each $D_t \in [0,1]$. Writing events in terms of D_t lets us use Lemma 3.1 to bound the probability that too many leaves are killed:

$$\begin{aligned}
\Pr\left[\ell + \sum_{t \in [\ell]} K_t \ge \prod_{i=1}^{d} b_i\right] &\le \Pr\left[\sum_{t \in [\ell]} K_t \ge 7 \max\left(\ell, \frac{1}{14}\prod_{j=1}^{d} b_j\right)\right] \\
&\le \Pr\left[\sum_{t \in [\ell]} D_t \ge 7 \max\left(\ell / \prod_{j=2}^{d} b_j, \frac{b_1}{14}\right)\right] \\
&\le \exp\left(-\frac{6^2}{2+6} \max\left(\ell / \prod_{j=2}^{d} b_j, \frac{b_1}{14}\right)\right) \\
&\le \exp\left(-\frac{9 b_1}{28}\right) \le 2^{b_1 \cdot \frac{9}{28 \ln 2}} \le 2^{\lceil 3 \log 1/\delta \rceil \frac{9}{28 \ln 2}} \le \delta .
\end{aligned}$$

□

In the preceding proof, we used the following:

Lemma 5.4. *Let $\hat{b}, \hat{w}, b, w$ be positive, and $\hat{b} \ge 1$. Then:*

$$\frac{\hat{b}}{\hat{w}}\left(\max\left(0, \frac{w}{b} - \max\left(\hat{b}\frac{w}{b} - \hat{w}, 0\right)\right)\right) \le 1 .$$

Proof of Lemma 5.4. If $\frac{\hat{b}}{\hat{w}} \le \frac{b}{w}$, then:

$$\frac{\hat{b}}{\hat{w}}\left(\max(0, \frac{w}{b} - \max(\hat{b}\frac{w}{b} - \hat{w}, 0))\right) \le \frac{\hat{b}}{\hat{w}}\frac{w}{b} \le 1 .$$

Otherwise, $\frac{\hat{b}}{\hat{w}} \ge \frac{b}{w}$, and:

$$\begin{aligned}
&\frac{\hat{b}}{\hat{w}}\left(\max\left(0, \frac{w}{b} - \max\left(\hat{b}\frac{w}{b} - \hat{w}, 0\right)\right)\right) \\
&\quad \le \frac{\hat{b}}{\hat{w}}\left(\frac{w}{b} - \left(\hat{b}\frac{w}{b} - \hat{w}\right)\right) = \frac{\hat{b}}{\hat{w}}\left(\frac{w}{b} - \hat{b}\left(\frac{w}{b} - \frac{\hat{w}}{\hat{b}}\right)\right) \\
&\quad \le \frac{\hat{b}}{\hat{w}}\left(\frac{w}{b} - 1\left(\frac{w}{b} - \frac{\hat{w}}{\hat{b}}\right)\right) = \frac{\hat{b}}{\hat{w}}\frac{\hat{w}}{\hat{b}} = 1 , \qquad \text{since } \hat{b} \ge 1 \text{ and } \frac{w}{b} - \frac{\hat{w}}{\hat{b}} \ge 0 .
\end{aligned}$$

□

The proof of the following lemma is a mostly straightforward calculation, which we defer to Appendix A.3.

Lemma 5.5. *Listing 3 uses*

$$O\left(\left\lceil \frac{(4\ell)^{2/(d-1)}}{(n/4)^{2/(d(d-1))}} \right\rceil (\log \ell)^2 + \min(\ell, \log 1/\delta) \log \ell\right) \tag{18}$$

bits of space, where $d = \min\left(\lceil \log \ell \rceil, \left\lfloor 2\frac{\log(n/4)}{\log(16\ell)} \right\rfloor\right)$. *A weaker upper bound on this is:*

$$O\left(\ell^{\frac{\log \ell}{\log n}} (\log \ell)^2 + \min(\ell, \log 1/\delta) \log \ell\right).$$

If $\ell \geq 4$ and $\ell \leq n/64$, then Lemma 5.5 and Lemma 5.3 together show that Listing 3 has error $\leq \delta$ and space usage as bounded by eq. (18). To handle the cases where $\ell < 4$ and $\ell > n/64$, one can instead use the simple deterministic algorithm for $\text{MIF}(n, \ell)$ from [Sto23], using only ℓ bits of space. As this is in fact less than the space upper bound from eq. (18), it follows that eq. (18) gives an upper bound on the space needed for a random tape, adversarially robust $\text{MIF}(n, \ell)$ algorithm for any setting of parameters. Formally:

Theorem 5.6. *There is a family of adversarially robust random tape algorithms, where for* $\text{MIF}(n, \ell)$ *the corresponding algorithm has* $\leq \delta$ *error and uses*

$$O\left(\left\lceil \frac{(4\ell)^{\frac{2}{d-1}}}{(n/4)^{\frac{2}{d(d-1)}}} \right\rceil (\log \ell)^2 + \min(\ell, \log \tfrac{1}{\delta}) \log \ell\right)$$

bits of space, where $d = \max\left(2, \min\left(\lceil \log \ell \rceil, \left\lfloor 2\frac{\log(n/4)}{\log(16\ell)} \right\rfloor\right)\right)$. *When* $\delta = 1/\text{poly}(n)$ *a (weakened) space bound is* $O\left(\ell^{\log_n \ell} (\log \ell)^2 + \log \ell \log n\right)$.

Remark. Listing 3 does not use the most optimal assignment of the parameters $b_d, \ldots, b_2$; constant-factor improvements in space usage are possible if one sets $b_d, \ldots, b_2$ to be roughly in an increasing arithmetic sequence, but this would make the analysis more painful.

Remark. In exchange for a constant factor space increase, one can adapt Listing 3 to produce an increasing sequence of output values. Similar adjustments can be performed for other MIF algorithms.

6 The Pseudo-Deterministic and Random Seed Lower Bounds

In this section, we prove a space lower bound for pseudo-deterministic streaming algorithms; in particular, for the most general (random oracle) type of them. See Section 2.4 for a high-level plan of the proof.

Let $\mathcal{A}$ be a random-oracle pseudo-deterministic algorithm for $\text{MIF}(n, \ell)$ using z bits of state, which has worst case failure probability $\delta \leq \frac{1}{3}$. Let $\Pi \colon [n]^\star \to [n]$ be the function giving the *canonical output* of $\mathcal{A}$ after processing a stream (as was defined in Section 1.1), and let $S = \Pi([n]^\ell)$ be the set of all canonical outputs at time ℓ (for this proof we can ignore outputs at times $< \ell$). Clearly $|S| \leq n$, and we will also prove (see Lemma 6.8) that $|S| \leq 2^{z+1}$. It is easy to see that $|S| \geq \ell + 1$ and that, since $\text{MIF}(n, \ell)$ is nontrivial, correct algorithms must have $z \geq 1$.

The main proof in this section only applies to algorithms with very low error (potentially as small as $1/n^{\Omega(\log n)}$). To ensure that we are working with an algorithm with error this small, we will first apply Lemma 3.3, using p independent instances of $\mathcal{A}$, where $p \geq 1$ is an integer chosen later. This will produce an ε-error algorithm $\mathcal{B}$ that uses zp bits of space, where $\varepsilon \leq (2\delta)^{p/30}$. Moreover, the canonical outputs of $\mathcal{B}$ will still be given by Π.

We may view $\mathcal{B}$ as a distribution over deterministic streaming algorithms; each such algorithm, obtained by fixing $\mathcal{B}$'s oracle random string, produces outputs according to a function of the form $A\colon [n]^\star \to [n]$. Since $\mathcal{B}$ is pseudo-deterministic, we have the following property:

$$\forall x \in [n]^{\le \ell}\colon \Pr_{A\sim\mathcal{B}}[A(x) \neq \Pi(x)] \le \varepsilon\,. \tag{19}$$

It is time to give the details of the key procedure FINDCOMMONOUTPUTS (abbreviated as FCO), outlined earlier in Section 2.4, that underpins our proof. The input stream positions, from 1 to ℓ, are split into $d = \Theta(\ell/(zp))$ consecutive intervals, of lengths $t_d, t_{d-1}, \ldots, t_1$, in order, where $t_d = t_{d-1} = \cdots = t_2 = \Theta(zp)$ and $t_1 = \ell - \sum_{k=2}^{d} t_k$; the constants are chosen such that $t_1 \ge \ell/2$. A call to FCO takes the form $\text{FCO}(B, C, x, k)$, where $B\colon [n]^\ell \to [n]$ is an output function, C is a random collection (matrix) of thresholds, $k \in [d]$, and $x \in [n]^{t_d+\cdots+t_{k+1}}$ is a stream prefix. (Notice that when $k = d$, we must have $x = \epsilon$.) By design FCO is recursive, bottoming out at $k = 1$, and always returns a subset of S of cardinality w_k, where

$$w_k := 2^{k-1}(t_1 + 1)\,. \tag{20}$$

The precise logic of FCO is given in Listing 4. Throughout this section, we use $\text{SEQS}(Z, t)$ to denote the set of all length-t *sorted sequences* of *distinct* elements of Z. Note that

$$|\text{SEQS}(Z, t)| = \binom{|Z|}{t}\,.$$

A key property of FCO is that for all $x \in [n]^{t_d+\cdots+t_{k+1}}$, if C is chosen uniformly at random from $[1,2)^{d\times\mathbb{N}}$, and $A \sim \mathcal{B}$, then w.h.p. $\text{FCO}(A, C, x, k)$ produces the same set as $\text{FCO}(\Pi, C, x, k)$. Another important property is that $\text{FCO}(\Pi, C, x, k)$ always produces w_k distinct elements in S without a failure (i.e., returns using Line 22, not Line 23). These properties are formally proved in Lemma 6.2, to follow. In particular, $\text{FCO}(\Pi, C, \epsilon, d)$ produces w_d distinct elements in S, showing that $|S| \ge w_d$. Since w_d is a function of z, combining this with upper bounds on $|S|$ lets us solve for a lower bound on z.

We elaborate on how we achieve the two key properties of FCO. First, to ensure $\text{FCO}(\Pi, C, x, k)$ equals $\text{FCO}(A, C, x, k)$ for random $A \sim \mathcal{B}$ and $C \in_R [1,2)^{d\times\mathbb{N}}$, we use a standard random threshold trick when computing the set P_h on Line 19. The recursive calls to $\text{FCO}(A, C, x \circ y, k-1)$ on Line 19 do not always match the outputs of $\text{FCO}(\Pi, C, x \circ y, k-1)$; as a result, the element frequency vector $f^{(h)}$ from Line 17 may have random noise when computed using A instead of using Π. If Line 19 used a fixed threshold value, then there would exist pseudo-deterministic MIF algorithms yielding element frequencies close to this threshold, where even low-magnitude noise could affect which elements are included in P_h. Using C to choose random threshold values that are independent of the choice of A ensures that most of the time, the noise has no influence on P_h; ultimately, ensuring that $\text{FCO}(\Pi, C, x, k)$ and $\text{FCO}(A, C, x, k)$ most likely produce the same output. More detail is given in the proof of Lemma 6.6.

Second, the design of Listing 4 ensures that $\text{FCO}(\Pi, C, \epsilon, d)$ actually produces a set of w_d possible outputs. Recall the $\text{AVOID}(m, a, b)$ communication problem described in Section 2.2. The output sets T_y of size w_{d-1} computed on Line 12, for each $y \in \text{SEQS}(S, t_d)$, will have a similar structure to the input and output sets for an AVOID protocol, in that the set T_y is typically disjoint from y, and in that the distribution of possible T_y values is limited (through the requirement that Π agrees with a mixture of functions corresponding to deterministic streaming algorithms). Note that we cannot just extract the "most common values" that occur in the sets T_y for $y \in \text{SEQS}(S, t_d)$, because if $t_d w_{d-1} \ll |S|$, it is possible that these are very concentrated; for example there could exist a set V of size w_{d-1} so that if $y \cap V = \varnothing$, then $T_y = V$, in which case an algorithm satisfying the first property can only reliably identify w_{d-1} outputs. Instead, we iteratively build up a set of common outputs, using the following observation. If the current set of common algorithm outputs Q_{h-1} is smaller than w_d, then the set P_h containing the most common elements of T_y for $y \in \text{SEQS}(Q_{h-1}, t_d)$ can not

Listing 4 The procedure to compute a set for Lemma 6.2

Let $t_1, \dots, t_d, w_1, \dots, w_d$ be integer parameters, and S the set of valid outputs

FINDCOMMONOUTPUTS(B, C, x, k) ▷ *abbreviated as* FCO(B,C,x,k)

▷ *Inputs: function* $B\colon [n]^\ell \to [n]$, *matrix* $C \in [1,2)^{d\times\mathbb{N}}$, *stream prefix* $x \in [n]^{t_k+\cdots+t_1}$ ◁

▷ *Output: a subset of* S *of size* w_k ◁

if $k = 1$ **then**
  $e_0 \leftarrow B(x \circ \langle 1, 1, \dots, 1\rangle)$
  for i in $1, \dots, t_1$ **do**
    $e_i \leftarrow B(x \circ \langle e_0, \dots, e_{i-1}, 1, \dots, 1\rangle)$
  if $e_0, \dots, e_{t_1}$ are all distinct **then**
    return $\{e_0, e_1, \dots, e_{t_1}\}$ ▷ *identify* w_1 *distinct possible outputs*
  return arbitrary subset of S of size w_1 (failure)
else
  for each $y \in \text{SEQS}([n], t_k)$ **do**
    $T_y \leftarrow$ FINDCOMMONOUTPUTS$(B, C, x \circ y, k-1)$ ▷ *note* $|T_y| = w_{k-1}$
  $Q_0 \leftarrow T_{\langle 1,2,\dots,t_k\rangle}$
  for h in $1, 2, 3, 4$ **do**
    ▷ *gather statistics and find common elements among the sets* T_y ◁
    for each $j \in S$ **do**
      $f_j^{(h)} \leftarrow |\{y \in \text{SEQS}(Q_{h-1}, t_k) : j \in T_y\}|$ ▷ *count frequencies*
    $\theta \leftarrow C_{k,h} w_{k-1} / (16|S|)$ ▷ *set random threshold*
    $P_h \leftarrow \left\{ j \in S : f_j^{(h)} \geq \theta \binom{|Q_{h-1}|}{t_k} \right\}$ ▷ *identify "sufficiently common" elements*
    $Q_h \leftarrow Q_{h-1} \cup P_h$
    if $|Q_h| \geq w_k$ **then**
      return the w_k smallest elements in Q_h
  return arbitrary subset of S of size w_k (failure)

be entirely contained by Q_{h-1}: if we did have $P_h \subseteq Q_{h-1}$, then one can show it is possible to construct an impossibly efficient protocol for AVOID$(|Q_{h-1}|, t_d, w_{d-1})$. Consequently, until for some value of h we have $|Q_h| \geq w_d$, it is possible to find a slightly larger Q_{h+1}. More detail is given in the proof of Lemma 6.4.

Before beginning the formal proof, let us set the various parameters precisely. Concretely, we set

$$p = \left\lceil \max\left(\sqrt{\frac{10\ell \log(64|S|)}{3z \log\frac{1}{2\delta}}}, \frac{30\log(64|S|)}{\log\frac{1}{2\delta}} \right) \right\rceil . \tag{21}$$

$$d = 1 + \left\lfloor \frac{\ell}{18zp} \right\rfloor , \tag{22}$$

$$t_d = t_{d-1} = \cdots = t_2 = \lceil 4\ln 2(zp+2) \rceil . \tag{23}$$

Recall also that

$$\varepsilon \leq (2\delta)^{p/30} . \tag{24}$$

We further define

$$\varepsilon_k := w_k (64|S|)^{k-1} \varepsilon , \tag{25}$$

which will upper-bound the probability that $\text{FCO}(A,C,x,k) \neq \text{FCO}(\Pi,C,x,k)$. We record some useful estimations in the next lemma.

Lemma 6.1. *With the above choices, $t_d = \cdots = t_2 \leq 9zp$, $t_1 \geq \ell/2$, and $\varepsilon \leq 1/(64|S|)^d$.*

Proof. The claim about the t_is holds since $z \geq 1$ and $p \geq 1$. Using this in eq. (22) gives $t_1 \geq \ell - (d-1)9zp \geq \ell/2$. We turn to proving the claim about ε. Recall that, by design, $\varepsilon \leq (2\delta)^{p/30}$.

The two branches of the maximum in eq. (21) ensure that:

$$p^2 \geq \frac{10\ell \log(64|S|)}{3z \log \frac{1}{2\delta}} \qquad \text{and} \qquad p \geq \frac{30 \log(64|S|)}{\log \frac{1}{2\delta}}. \tag{26}$$

Because $d \leq \max(1, \frac{\ell}{9zp})$, it suffices to prove that $\varepsilon \leq 1/(64|S|)$ and that $\varepsilon \leq 1/(64|S|)^{\ell/(9zp)}$. Now,

$$\log \frac{1}{\varepsilon} \underset{\text{Lemma 3.3}}{\geq} \frac{p}{30} \log \frac{1}{2\delta} \underset{\text{eq. (26)}}{\geq} \frac{1}{30p} \frac{10\ell \log(64|S|)}{3z} = \frac{\ell}{9zp} \log(64|S|);$$
$$\log \frac{1}{\varepsilon} \underset{\text{Lemma 3.3}}{\geq} \frac{p}{30} \log \frac{1}{2\delta} \underset{\text{eq. (26)}}{\geq} \log(64|S|).$$

□

6.1 Common Outputs Behave Canonically

We now come to the central lemma in the proof, which asserts that the set of common outputs is likely the same for the canonical function Π as it is for a random draw $A \sim \mathcal{B}$. It also asserts two other key properties of FCO. The lemma can be thought of as a "proof of correctness" of FCO.

Lemma 6.2. *Let $k \in [d]$ and $x \in [n]^{t_d + \cdots + t_{k+1}}$. Then* FCO *satisfies the following properties.*

1. $\Pr_{A \sim \mathcal{B}, C \in_R [1,2)^{d \times \mathbb{N}}}[\text{FCO}(A,C,x,k) = \text{FCO}(\Pi,C,x,k)] \geq 1 - \varepsilon_k$.
2. *For all $C \in [1,2)^d$, the set $\text{FCO}(\Pi,C,x,k)$ is disjoint from x and a subset of S.*
3. *For all $A\colon [n]^\ell \to [n]$ and $C \in [1,2)^d$, $\text{FCO}(A,C,x,k)$ outputs a set of size w_k.*

Proof. The proof is by induction on k, spread over the next few lemmas. The case $k = 1$ is handled in Lemma 6.3. For the induction step, the heart of the argument, which invokes a communication lower bound for AVOID, is given in Lemma 6.4. Following this, Lemma 6.5 establishes the latter two claims in the lemma and Lemma 6.6 establishes the first claim. □

Lemma 6.3. *Lemma 6.2 holds for $k = 1$.*

Proof. Let $e_0, \ldots, e_{t_1}$ be the values of the variables on Lines 4 to 6 of Listing 4 when $\text{FCO}(\Pi,C,x,1)$ is called; note that these do not depend on C. For $i \in \{0,\ldots,t_1\}$, define sequences $s_i = \langle e_0, \ldots, e_{i-1}, 1, \ldots, 1\rangle$, so that $s_0 = \langle 1,1,\ldots,1\rangle$, and $s_{t_1} = \langle e_0, \ldots, e_{t_1-1}\rangle$. If, for all $i \in \{0,\ldots,t_1\}$, we have $A(x \circ s_i) = \Pi(x \circ s_i)$, the value of $\text{FCO}(A,C,x,1)$ will exactly match $\text{FCO}(\Pi,C,x,1)$. By a union bound,

$$\Pr_{A \sim \mathcal{B}, C}[\text{FCO}(A,C,x,k) \neq \text{FCO}(\Pi,C,x,k)] \leq \sum_{i=0}^{t_1} \Pr_{A \sim \mathcal{B}}[A(x \circ s_i) \neq \Pi(x \circ s_i)] \underset{\text{eq. (19)}}{\leq} (t_1 + 1)\varepsilon = \varepsilon_1.$$

Because Π is the canonical output function for a protocol for MIF, for any $z \in [n]^\ell$, we have $\Pi(z) \notin z$. Consequently, each $e_i = \Pi(x \circ \langle e_0, \ldots, e_{i-1}, 1, \ldots, 1\rangle)$ is neither contained in x nor by $\{e_0,\ldots,e_{i-1}\}$; thus $\{e_0,\ldots,e_{t_1}\}$ has size $t_1 + 1 = w_1$ and is disjoint from x. □

Lemma 6.4. *Let $x \in [n]^{t_d+\cdots+t_{k+1}}$. When computing* $\textsc{fco}(\Pi, C, x, k)$*, in the hth loop iteration, if* $|Q_{h-1}| < w_k$*, then* $|P_h \setminus Q_{h-1}| \geq \left\lceil \frac{1}{4} w_{k-1} \right\rceil$*. Consequently, the algorithm will return using Line 22, not Line 23.*

Proof. Assume for sake of contradiction that $|Q_{h-1}| < w_k$ and $|P_h \setminus Q_{h-1}| \leq \left\lfloor \frac{1}{4} w_{k-1} \right\rfloor$. Then we can use the algorithm $\mathcal{A}$ to implement a protocol for the one-way communication problem $\textsc{avoid}(|Q_{h-1}|, t_k, \left\lceil \frac{1}{2} w_{k-1} \right\rceil)$, with $\leq \frac{1}{2}$ probability of error.

We assume without loss of generality that $Q_{h-1} = [|Q_{h-1}|]$; if not, relabel coordinates so that this holds. In the protocol, after Alice is given a subset $W \subseteq Q_{h-1}$ with $|W| = t_k$, they construct a sequence $v = x \circ \textsc{sort}(W)$ in $[n]^{t_d+\cdots+t_k}$. Then Alice uses public randomness to instantiate an instance E of $\mathcal{A}$; inputs the sequence v to E; and sends the new state of E to Bob, using a zp-bit message. As Bob shares the public randomness, they can use this state to evaluate the output of the algorithm on any continuation of the stream. In particular, Bob can evaluate the algorithm for any possible suffix, to produce a function $\tilde{A}_{x \circ \textsc{sort}(W)} : [n]^{t_{k-1}+\cdots+t_1} \to [n]$; Bob then samples a random $C \in [1,2)^{d \times \mathbb{N}}$, and computes $V = \textsc{fco}(\tilde{A}_{x \circ \textsc{sort}(W)}, C, k-1)$, which is a subset of S. If $|V \cap Q_{h-1}| \geq \left\lceil \frac{1}{2} w_{k-1} \right\rceil$, Bob outputs the smallest $\left\lceil \frac{1}{2} w_{k-1} \right\rceil$ entries of $V \cap Q_{h-1}$. Otherwise, Bob outputs an arbitrary set of size $\left\lceil \frac{1}{2} w_{k-1} \right\rceil$.

First, we observe that for any value of $\textsc{sort}(W)$, the distribution of $\tilde{A}_{x \circ \textsc{sort}(W)}$ is exactly the same as the distribution of $A_{x \circ \textsc{sort}(W)}$, when A is drawn from $\mathcal{B}$; this follows because for a fixed setting of the oracle random string of the algorithm, it behaves deterministically.

Applying Lemma 6.2 at $k-1$, we observe that for any $W \in \binom{Q_{h-1}}{t_k}$,

$$\Pr[\textsc{fco}(\tilde{A}_{x \circ \textsc{sort}(W)}, C, k-1) = \textsc{fco}(\Pi, C, x \circ \textsc{sort}(W), k-1)] \geq 1 - \varepsilon_{k-1} \geq \frac{3}{4}.$$

Furthermore, we are guaranteed that $\textsc{fco}(\Pi, C, x \circ \textsc{sort}(W), k-1)$ has size w_{k-1} and is disjoint from W.

We now bound the probability, over a uniformly random $y \in \textsc{seqs}(Q_{h-1}, t_k)$, that $|\textsc{fco}(\Pi, C, x \circ y, k-1) \cap Q_{h-1}| < \left\lceil \frac{1}{2} w_{k-1} \right\rceil$. Define $T_y = \textsc{fco}(\Pi, C, x \circ y, k-1)$ and, for each $j \in S$, $f_j^{(h)}$, as in Listing 4. In particular, we have:

$$\begin{aligned}
\Pr_{y,C}\left[|T_y \cap Q_{h-1}| < \left\lceil \frac{1}{2} w_{k-1} \right\rceil \right] &= \Pr_{y,C}\left[|T_y \setminus Q_{h-1}| > \left\lfloor \frac{1}{2} w_{k-1} \right\rfloor \right] \\
&\leq \Pr_{y,C}\left[|T_y \setminus P_h \setminus Q_{h-1}| > \left\lfloor \frac{1}{2} w_{k-1} \right\rfloor - \left\lfloor \frac{1}{4} w_{k-1} \right\rfloor \right] \qquad (27) \\
&\leq \Pr_{y,C}\left[|T_y \setminus P_h| \geq \frac{1}{4} w_{k-1} \right].
\end{aligned}$$

(The inequality on eq. (27) follows since we assumed $|P_h \setminus Q_{h-1}| \leq \left\lfloor \frac{1}{4} w_{k-1} \right\rfloor$.) Note that:

$$\sum_{j \notin P_h} f_j^{(h)} = \sum_{y \in \textsc{seqs}(Q_{h-1}, t_k)} |T_y \setminus P_h| \geq \frac{1}{4} w_{k-1} \left| \left\{ y \in \textsc{seqs}(Q_{h-1}, t_k) : |T_y \setminus P_h| \geq \frac{1}{4} w_{k-1} \right\} \right|. \qquad (28)$$

Using the fact that y is uniformly distributed over $\textsc{seqs}(Q_{h-1}, t_k)$, gives:

$$\begin{aligned}
\Pr_{y,C}\left[|T_y \setminus P_h| \geq \frac{1}{4}w_{k-1}\right] &= \mathbb{E}_C \frac{\left|\left\{y \in \textsc{seqs}(Q_{h-1}, t_k) : |T_y \setminus P_h| \geq \frac{1}{4}w_{k-1}\right\}\right|}{\binom{|Q_{h-1}|}{t_k}} \\
&\leq \mathbb{E}_C \frac{\sum_{j \notin P_h} f_j^{(h)}}{\frac{1}{4}w_{k-1}\binom{|Q_{h-1}|}{t_k}} && \text{by eq. (28)} \\
&\leq \mathbb{E}_C \frac{(|S| - |P_h|)\frac{C_{k,h}w_{k-1}}{16|S|}\binom{|Q_{h-1}|}{t_k}}{\frac{1}{4}w_{k-1}\binom{|Q_{h-1}|}{t_k}} && \text{by definition of } P_h \\
&= \mathbb{E}_C \frac{C_{k,h}}{4}\frac{|S| - |P_h|}{|S|} \leq \frac{1}{2}. && \text{since } |P_h| \geq 0,\ C_{k,h} \leq 2
\end{aligned}$$

Thus the probability that $|T_y \cap Q_{h-1}| < \lceil \frac{1}{2}w_{k-1} \rceil$ holds is $\leq 1/2$. Since Bob only gives an incorrect output when this happens or when $\textsc{fco}(\tilde{A}_{x \circ \textsc{sort}(W)}, C, k-1) \neq \textsc{fco}(\Pi, C, x \circ \textsc{sort}(W), k-1)$, it follows by a union bound that the total failure probability is $\leq \frac{1}{2} + \frac{1}{4} \leq \frac{3}{4}$.

Consequently, the protocol implementation has $\leq \frac{3}{4}$ error when inputs are drawn from the uniform distribution over $\binom{Q_{h-1}}{t_k}$; by Theorem 3.4, we obtain a lower bound on the required message length, giving

$$zp > \frac{t_k\lceil \frac{1}{2}w_{k-1} \rceil}{|Q_{h-1}|\ln 2} + \log(1 - 3/4) \geq \frac{t_k w_{k-1}}{|Q_{h-1}| \cdot 2\ln 2} - 2.$$

Rearranging this slightly and using integrality of $|Q_{h-1}|$ gives:

$$|Q_{h-1}| \geq \left\lceil \frac{t_k w_{k-1}}{2\ln 2(zp+2)} \right\rceil = \left\lceil \frac{\lceil 4\ln 2(zp+2) \rceil}{2\ln 2(zp+2)} w_{k-1} \right\rceil \geq 2w_{k-1} = w_k,$$

but as $|Q_{h-1}| < w_k$, this implies $w_k < w_k$, which is a contradiction; this proves that the assumption $|P_h \setminus Q_{h-1}| \leq \frac{1}{4}w_{k-1}$ must have been invalid.

Finally, we observe that since, in each iteration of the loop on Lines 14 to 22, $|Q_h| = |Q_{h-1} \cup P_h| = |Q_{h-1}| + |P_h \setminus Q_{h-1}| \geq |Q_{h-1}| + \lceil \frac{1}{4}w_{k-1} \rceil$, and we initially have $|Q_0| = w_{k-1}$, the size of Q_h (assuming we haven't returned yet) must be $\geq w_{k-1}(1 + h/4)$. By the last loop iteration (with $h = 4$), we will have $|Q_h| \geq 2w_{k-1} = w_k$. □

Lemma 6.5. *For $k > 1$, $x \in [n]^{t_d + \cdots + t_{k+1}}$, $\textsc{fco}(\Pi, C, x, k)$ is disjoint from x and a subset of S; and for all A, C, k, $\textsc{fco}(A, C, x, k)$ outputs a set of size w_k.*

Proof. By Lemma 6.2 at $k-1$, the sets $T_{A, x \circ y}$ chosen on Line 12 are always subsets of S and disjoint from $x \circ y$, and hence disjoint from x. Per Lemma 6.4, FindCommonOutputs will return a subset of Q_h using Line 22, where h is the last loop iteration number. Each element of Q_h was either in $T_{A, x \circ \langle 1,2,\ldots,t_k \rangle}$ (and hence also in S) or was in $P_{h'}$ for some $h' \leq h$. Note that $P_{h'}$ only contains integers j for which $f_j^{(h')} > 0$; i.e., which were contained in one of the sets $(T_{A, x \circ y})_{y \in \textsc{seqs}(Q_{h'-1}, t_k)}$, and are thereby also in S. Consequently, the set returned is contained in S, which implies $|S| \geq w_k$.

Calls to $\textsc{fco}(A, C, x, k)$ will either output through Line 22 (where the size of the set has been checked by the pseudocode) or through Line 23 (where a subset of size w_k must exist, because we know $|S| \geq w_k$). □

Lemma 6.6. *For $k > 1$, and all $x \in [n]^{t_d + \cdots + t_{k+1}}$,*

$$\Pr_{A \sim \mathcal{D}, C}[\textsc{fco}(A, C, x, k) \neq \textsc{fco}(\Pi, C, x, k)] \leq \varepsilon_k.$$

Proof. The proof of the lemma follows from the observation that, when computing $\text{FCO}(A,C,x,k)$, even if a fraction of the recursive calls to $\text{FCO}(A,C,x\circ y,k-1)$ produced incorrect outputs, the values for Q_0 and $(P_h)_{h\geq 1}$ will likely match those computed when $\text{FCO}(\Pi,C,x,k)$ is called.

Henceforth, we indicate variables from the computation of $\text{FCO}(\Pi,C,x,k)$ without a tilde, and variables from the computation of $\text{FCO}(A,C,x,k)$ with a tilde. For example, $f_j^{(h)}$ is computed using $B=\Pi$, while $\tilde{f}_j^{(h)}$ is computed using $B=A$. We also define

$$\hat{f}_j^{(h)} = \left|\{y\in \text{SEQS}(Q_{h-1},t_k) : j\in T_{A,y}\}\right|$$
$$\hat{P}_h = \left\{ j\in S : \hat{f}_j^{(h)} \geq \frac{C_{k,h}w_{k-1}}{16|S|}\left|\binom{Q_{h-1}}{t_k}\right|\right\};$$

that is, $\hat{f}_j^{(h)}$ and $\hat{P}_h$ are the values that would be computed by $\text{FCO}(A,C,x,k)$ if the set Q_{h-1} was used instead of the set $\widetilde{Q}_{h-1}$.

Say $\text{FCO}(\Pi,C,x,k)$ returns from the loop at iteration $h^\star$. The output of $\text{FCO}(A,C,x,k)$ will equal $\text{FCO}(\Pi,C,x,k)$ if $Q_0=\widetilde{Q}_0$ and for all $h\in[h^\star]$, we have $P_h=\hat{P}_h$. (If this occurs, then as $Q_0=\widetilde{Q}_0$, $\hat{P}_1=\widetilde{P}_1$, so $Q_1=Q_0\cup P_1=\widetilde{Q}_0\cup\widetilde{P}_1=\widetilde{Q}_1$, and as $Q_1=\widetilde{Q}_1$, $\hat{P}_2=\widetilde{P}_2$, and so on.) By Lemma 6.2 at $k-1$, the probability that $Q_0\neq\widetilde{Q}_0$ is $\leq\varepsilon_{k-1}$. Consider a specific $h\in[h^\star]$; the only way in which $\hat{P}_h\neq P_h$ is if there is some $j\in S$ for which $f_j^{(h)}$ and $\hat{f}_j^{(h)}$ are on opposite sides of the threshold $\frac{C_{k,h}w_{k-1}}{16|S|}\left|\binom{Q_{h-1}}{t_k}\right|$.

Let λ_h be the random variable indicating the fraction of $y\in\text{SEQS}(Q_{h-1},t_k)$ for which $T_{A,x\circ y}\neq T_{\Pi,x\circ y}$. Note that the values $T_{A,x\circ y}$ are functions of the random variable A and of $C_{k',h}$ for $k'<k, h\in\mathbb{N}$; in particular $T_{A,x\circ y}$ is independent of $(C_{k,h})_{h\in\mathbb{N}}$. By Lemma 6.2 at $k-1$, $\Pr[T_{A,x\circ y}\neq T_{\Pi,x\circ y}]\leq\varepsilon_{k-1}$, which implies $\mathbb{E}\lambda_h\leq\varepsilon_{k-1}$.

Fix a particular setting of A and $(C_{k',h})_{k'<k,h\in\mathbb{N}}$. Since each set $T_{A,x\circ y}$ contributes 1 unit to each of w_{k-1} variables $\hat{f}_j^{(h)}$:

$$\sum_{j\in S}\left|f_j^{(h)}-\hat{f}_j^{(h)}\right| \leq w_{k-1}\left|\{y\in\text{SEQS}(Q_{h-1},t_k) : T_{A,x\circ y}\neq T_{\Pi,x\circ y}\}\right| = w_{k-1}\lambda_h\binom{Q_{h-1}}{t_k}.$$

Let F be the set of possible values in $[1,2)$ for $C_{k,h}$ for which $P_h\neq\hat{P}_h$; this is a union of intervals corresponding to each pair $\left(f_j^{(h)},\hat{f}_j^{(h)}\right)$, for $j\in S$. A given value c is bad for j if

$$f_j^{(h)} < \frac{cw_{k-1}}{16|S|}\binom{|Q_{h-1}|}{t_k} \leq \hat{f}_j^{(h)}; \qquad \text{equivalently:} \qquad c\in\left(\frac{16|S|f_j^{(h)}}{w_{k-1}\binom{|Q_{h-1}|}{t_k}}, \frac{16|S|\hat{f}_j^{(h)}}{w_{k-1}\binom{|Q_{h-1}|}{t_k}}\right],$$

and similarly in the case where $\hat{f}_j^{(h)}<f_j^{(h)}$. The measure of F is:

$$\leq\sum_{j\in S}\frac{16|S|}{w_{k-1}\binom{|Q_{h-1}|}{t_k}}|\hat{f}_j^{(h)}-f_j^{(h)}| \leq \frac{16|S|}{w_{k-1}}w_{k-1}\lambda_h = 16|S|\lambda_h.$$

This upper bounds the probability that $C_{k,h}\in F$ and $P_h\neq\hat{P}_h$. We then have:

$$\Pr[P_h\neq\hat{P}_h] = \mathbb{E}_{A,(C_{k',h})_{k'<k}}\Pr[C_{k,h}\in F] \leq \mathbb{E}_{A,(C_{k',h})_{k'<k}}(16|S|\lambda_h) = 16|S|\varepsilon_{k-1}.$$

By a union bound, the probability that $Q_0\neq\widetilde{Q}_0$ or $P_h\neq\hat{P}_h$ for any $h\leq h^\star$ is

$$\leq\varepsilon_{k-1}+h^\star 16|S|\varepsilon_{k-1}\leq(1+4\cdot 16|S|)\varepsilon_{k-1}\leq\frac{64|S|w_k}{w_{k-1}}\varepsilon_{k-1}.$$

Thus $\Pr[\text{FCO}(A,C,x,k)\neq\text{FCO}(\Pi,C,x,k)]\leq\frac{64|S|w_k}{w_{k-1}}\varepsilon_{k-1}=\varepsilon_k$. □

We have thus completed the proof of Lemma 6.2, establishing the key properties of FCO. It is time to use them to derive our lower bound.

6.2 Obtaining a Pseudo-Deterministic Lower Bound

Lemma 6.7. *We have:*

$$z \geq \frac{\ell}{\log \frac{2|S|}{\ell}} \min\left(\frac{1}{36}, \frac{\log \frac{1}{2\delta}}{17280 \log(64|S|) \log \frac{2|S|}{\ell}}\right).$$

Proof. A consequence of Lemma 6.2 is that $\text{FCO}(\Pi, C, d)$ will output a set of size w_d which is a subset of S. This shows that $|S| \geq w_d$. Now, from the definition of w_d, it follows that

$$|S| \geq w_d = 2^{d-1}(t_1+1) > 2^{d-1} t_1 \geq 2^{d-1} \frac{\ell}{2} \qquad \Longrightarrow \qquad \log \frac{2|S|}{\ell} \geq d - 1 = \left\lfloor \frac{\ell}{18zp} \right\rfloor.$$

Since $|S| \geq \ell + 1$, the left hand side $\log \frac{2|S|}{\ell} > 1$, so using the inequality $x/2 \leq \max(\lfloor x \rfloor, 1)$ gives:

$$\frac{\ell}{36zp} \leq \log \frac{2|S|}{\ell} \qquad \Longrightarrow \qquad z \geq \frac{\ell}{36 \log \frac{2|S|}{\ell}} \cdot \frac{1}{p}.$$

Next, we expand the definition of p (see eq. (21)), eliminating the ceiling using the inequality $\lceil x \rceil \leq \max(1, 2x)$:

$$z \geq \frac{\ell}{36 \log \frac{2|S|}{\ell}} \min\left(1, \frac{1}{2}\sqrt{\frac{3z \log \frac{1}{2\delta}}{10\ell \log(64|S|)}}, \frac{1}{2} \cdot \frac{\log \frac{1}{2\delta}}{30 \log(64|S|)}\right).$$

We have two cases: if the left or right side of the minimum is smallest, then:

$$z \geq \min\left(\frac{\ell}{36 \log \frac{2|S|}{\ell}}, \frac{\ell \log \frac{1}{2\delta}}{2160 \log \frac{2|S|}{\ell} \log(64|S|)}\right), \tag{29}$$

while otherwise, if the center is smallest, we get:

$$z^2 \geq \frac{1}{4} \frac{\ell^2}{(36 \log \frac{2|S|}{\ell})^2} \cdot \frac{3z \log \frac{1}{2\delta}}{10\ell \log(64|S|)} \qquad \Longrightarrow \qquad z \geq \frac{\ell \log \frac{1}{2\delta}}{17280 (\log \frac{2|S|}{\ell})^2 \log(64|S|)}.$$

As $\log \frac{2|S|}{\ell} \geq 1$, this is smaller than the right minimum branch of eq. (29), so the common lower bound for all cases is:

$$z \geq \min\left(\frac{\ell}{36 \log \frac{2|S|}{\ell}}, \frac{\ell \log \frac{1}{2\delta}}{17280 (\log \frac{2|S|}{\ell})^2 \log(64|S|)}\right). \qquad \square$$

Lemma 6.8. *We have* $|S| < 2^{z+1}$.

Proof. For each $a \in S$, let $x_a \in \Pi^{-1}(a)$. One can use $\mathcal{A}$ to provide a randomized $\leq \delta$-error, z-bit encoding of the elements in S. Using public randomness, encoder and decoder choose the oracle random string for $\mathcal{A}$. Each $a \in S$ is encoded by sending x_a to $\mathcal{A}$ and outputting the algorithm state σ. To decode, given a state σ, one evaluates the output of $\mathcal{A}$ at state σ. Using the minimax principle, one can prove that the randomized encoding requires $\geq \log((1-\delta)|S|)$ bits of space, which implies $2^z \geq (1-\delta)|S|$. Since $\delta \leq \frac{1}{3}$, it follows $s \leq \frac{3}{2} 2^z < 2^{z+1}$. $\square$

We now establish the main result.

Theorem 6.9. *Pseudo-deterministic δ-error random oracle algorithms for* $\text{MIF}(n,\ell)$ *require*

$$\Omega\left(\min\left(\frac{\ell}{\log\frac{2n}{\ell}}+\sqrt{\ell},\frac{\ell\log\frac{1}{2\delta}}{(\log\frac{2n}{\ell})^2\log n}+\left(\ell\log\frac{1}{2\delta}\right)^{1/4}\right)\right)$$

bits of space when $\delta\le\frac{1}{3}$. In particular, when $\delta=1/\text{poly}(n)$ and $\ell=\Omega(\log n)$, this is:

$$\Omega\left(\frac{\ell}{(\log\frac{2n}{\ell})^2}+(\ell\log n)^{1/4}\right).$$

Proof sketch. Using Lemma 6.8 and the fact that $S\subseteq[n]$, we obtain $|S|\le\min(n,2^{z+1})$. The theorem follows by combining this bound with the inequality of Lemma 6.7, and for each of four cases corresponding to different branches of min and max, solving to find a lower bound on z. The full proof with calculations is given in Appendix A.4. □

Remark. For $\delta\le 2^{-\ell}$, Theorem 6.9 reproduces the deterministic algorithm space lower bound for $\text{MIF}(n,\ell)$ from [Sto23] within a constant factor.

6.3 Implications for Adversarially Robust Random Seed Algorithms

The following result, paraphrased from [Sto23] relates the random seed adversarially robust space complexity with the pseudo-deterministic space complexity.

Theorem 6.10 ([Sto23])**.** *Let $S^{PD}_{1/3}(n,\ell)$ give a space lower bound for a pseudo-deterministic algorithm for* $\text{MIF}(n,\ell)$ *with error $\le 1/3$. Then an adversarially robust random seed algorithm with error $\delta\le\frac{1}{6}$, if it uses z bits of space, must have $z\ge S^{PD}_{1/3}(n,\lfloor\frac{\ell}{2z+2}\rfloor)$.*

Theorem 6.11. *Adversarially robust random seed algorithms for* $\text{MIF}(n,\ell)$ *with error $\le\frac{1}{6}$ require space:*

$$\Omega\left(\frac{\ell^2}{n}+\sqrt{\ell/(\log n)^3}+\ell^{1/5}\right).$$

This follows by combining Theorem 6.10, Theorem 6.9, Theorem 3.5, and performing some algebra; a proof is given in Appendix A.4.

Acknowledgements

We thank Jonathan Conroy for helpful feedback on a earlier draft of this paper. We also thank Omri Ben-Eliezer for the sampling versus sketching interpretation of a portion of our results (noted after Result 3).

References

[ABJ+22] Miklós Ajtai, Vladimir Braverman, T.S. Jayram, Sandeep Silwal, Alec Sun, David P. Woodruff, and Samson Zhou. The white-box adversarial data stream model. In *Proc. 41st ACM Symposium on Principles of Database Systems*, pages 15–27, 2022.

[ACGS23] Sepehr Assadi, Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Coloring in graph streams via deterministic and adversarially robust algorithms. In *Proc. 42nd ACM Symposium on Principles of Database Systems*, pages 141–153, 2023.

[BEEO22] Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. Adversarially robust streaming via dense-sparse trade-offs. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 214–227, 2022.

[BJWY20] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In *Proc. 39th ACM Symposium on Principles of Database Systems*, page 63–80, 2020.

[BKKS23] Vladimir Braverman, Robert Krauthgamer, Aditya Krishnan, and Shay Sapir. Lower bounds for pseudo-deterministic counting in a stream. *arXiv preprint arXiv:2303.16287*, 2023.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BY20] Omri Ben-Eliezer and Eylon Yogev. The adversarial robustness of sampling. In *Proc. 39th ACM Symposium on Principles of Database Systems*, pages 49–62. ACM, 2020.

[CGS22] Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Adversarially robust coloring for graph streams. In *Proc. 13th Conference on Innovations in Theoretical Computer Science*, pages 37:1–37:23, 2022.

[Fei19] Uriel Feige. A randomized strategy in the mirror game. *arXiv preprint arXiv:1901.07809*, 2019.

[GGMW20] Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. Pseudo-Deterministic Streaming. In *Proc. 20th Conference on Innovations in Theoretical Computer Science*, volume 151, pages 79:1–79:25, 2020.

[GGS23] Ofer Grossman, Meghal Gupta, and Mark Sellke. Tight space lower bound for pseudo-deterministic approximate counting. *arXiv preprint arXiv:2304.01438*, 2023.

[GS18] Sumegha Garg and Jon Schneider. The Space Complexity of Mirror Games. In *Proc. 10th Conference on Innovations in Theoretical Computer Science*, pages 36:1–36:14, 2018.

[HKM+20] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri 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.

[HW13] Moritz Hardt and David P. Woodruff. How robust are linear sketches to adaptive inputs? In *Proc. 45th Annual ACM Symposium on the Theory of Computing*, pages 121–130, 2013.

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

[JP83] Kumar Joag-Dev and Frank Proschan. Negative association of random variables, with applications. *Ann. Stat.*, 11(1):286–295, 1983.

[JW23] Rajesh Jayaram and David P Woodruff. Towards optimal moment estimation in streaming and distributed models. *ACM Trans. Alg.*, 19(3):1–35, 2023.

[KMNS21] Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri 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*, volume 12827 of *Lecture Notes in Computer Science*, pages 94–121. Springer, 2021.

[MN22a] Roey Magen and Moni Naor. Mirror games against an open book player. In *11th International Conference on Fun with Algorithms (FUN 2022)*, volume 226, pages 20:1–20:12, 2022.

[MN22b] Boaz Menuhin and Moni Naor. Keep that card in mind: Card guessing with limited memory. In *Proc. 13th Conference on Innovations in Theoretical Computer Science*, pages 107:1–107:28, 2022.

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Inform. Process. Lett.*, 39(2):67–71, 1991.

[Nis90] Noam Nisan. Pseudorandom generators for space-bounded computation. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 204–212, 1990.

[Nis93] Noam Nisan. On read once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107(1):135–144, 1993.

[Sto23] Manuel Stoeckl. Streaming algorithms for the missing item finding problem. In *Proc. 34th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 793–818, 2023. Full version at arXiv:2211.05170v1.

[WZ22] David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science*, pages 1183–1196, 2022.

A Appendix

A.1 Proofs of Useful Lemmas

Here we provide proofs of the results in Section 3.1 for which we haven't found an external source.

Proof of Lemma 3.1. The proof is modeled off that in [Sto23], which only addresses one direction. It is a straightforward blend of standard proofs of the Chernoff bound and of Azuma's inequality.

First, the $\geq$ direction. Choose, with foresight, $z = \ln(1+\alpha)$.

$$\begin{aligned}
&\Pr\left[\sum_{i=1}^{t} X_i \geq (1+\alpha)\sum_{i=1}^{t} p_i\right] \\
&= \Pr\left[\exp(z\sum_{i=1}^{t} X_i) \geq \exp(z(1+\alpha)\sum_{i=1}^{t} p_i)\right] \\
&\leq \frac{\mathbb{E}\exp(z\sum_{i=1}^{t} X_i)}{\exp(z(1+\alpha)\sum_{i=1}^{t} p_i)} \\
&\leq \frac{\mathbb{E}[e^{zX_1}\mathbb{E}[e^{zX_2}\ldots\mathbb{E}[e^{zX_t}|X_1,\ldots,X_{t-1}]\ldots|X_1]]}{\exp(z(1+\alpha)\sum_{i=1}^{t} p_i)}.
\end{aligned}$$

The innermost term $\mathbb{E}[e^{zX_t}|X_1,\ldots,X_{t-1}]$ is, by convexity of e^z, $\leq p_t e^z + (1-p_t) \leq e^{p_t(e^z-1)}$; after applying this upper bound, we can factor it out and bound the X_{t-1} term, and so on. Thus we continue the chain of inequalities to get:

$$\leq \frac{\exp\left((e^z-1)\sum_{i=1}^t p_i\right)}{\exp\left(z(1+\alpha)\sum_{i=1}^t p_i\right)} = \exp\left(-((1+\alpha)\ln(1+\alpha)-\alpha)\sum_{i=1}^t p_i\right).$$

For the other direction, set $z = \ln(1-\alpha)$, which is < 0. This time, $\mathbb{E}[e^{zX_t}|X_1,\ldots,X_{t-1}] \leq p_t e^z + (1-p_t)$ because p_t is a *lower bound* for $\mathbb{E}[X_t|X_1,\ldots,X_{t-1}]$, and z is negative. That $p_t e^z + (1-p_t) \leq e^{p_t(e^z-1)}$ still holds for negative z, so:

$$\begin{aligned}\Pr\left[\sum_{i=1}^t X_i \leq (1-\alpha)\sum_{i=1}^t p_i\right] &= \Pr\left[\exp(z\sum_{i=1}^t X_i) \geq \exp(z(1-\alpha)\sum_{i=1}^t p_i)\right] \\ &\leq \ldots \leq \frac{\exp\left((e^z-1)\sum_{i=1}^t p_i\right)}{\exp\left(z(1-\alpha)\sum_{i=1}^t p_i\right)} \\ &= \exp\left(-((1-\alpha)\ln(1-\alpha)+\alpha)\sum_{i=1}^t p_i\right). \qquad \square\end{aligned}$$

Proof of Lemma 3.3. For each $i \in [p]$, let Y_i be the random indicator variable for the event that $X_i \neq v$. Let $\alpha = \frac{1}{2\delta} - 1$.The probability that v is not the most common element can be bounded by the probability that it is the not the majority element; by a Chernoff bound, this is:

$$\begin{aligned}\Pr[\sum_{i\in[p]} Y_i \geq \frac{1}{2}p] = \Pr[\sum_{i\in[p]} Y_i \geq (1+\alpha)\delta p] &\leq \exp\left(-((1+\alpha)\ln(1+\alpha)-\alpha)\delta p\right) \\ &\leq \exp\left(-0.073((1+\alpha)\ln(1+\alpha))\delta p\right) \qquad \text{since } \alpha \geq 1/6 \\ &\leq \exp\left(-\frac{0.073}{2\delta}\ln\frac{1}{2\delta}\delta p\right) = (2\delta)^{0.036p}. \qquad \square\end{aligned}$$

A.2 Mechanical Proofs for Section 4

Proof of Lemma 4.7. Let x be the left hand side of Eq. 14. The left branch of the $\min(\cdot,\cdot)$ terms in Eq. 14 is actually unnecessary. For any integer $\lambda \geq 2$, say that

$$\left(\frac{\ell^{\lambda+1}}{n}\right)^{\frac{2}{\lambda^2+3\lambda-2}} = \max_{k\in\mathbb{N}}\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}.$$

Then in particular,

$$\left(\frac{\ell^{\lambda+1}}{n}\right)^{\frac{2}{\lambda^2+3\lambda-2}} \geq \left(\frac{\ell^{(\lambda-1)+1}}{n}\right)^{\frac{2}{(\lambda-1)^2+3(\lambda-1)-2}} \geq \left(\frac{\ell^{\lambda}}{n}\right)^{\frac{2}{\lambda^2+\lambda-4}},$$

which implies

$$n^{2\lambda+2} = n^{(\lambda^2+3\lambda-2)-(\lambda^2+\lambda-4)} \geq \ell^{\lambda\cdot(\lambda^2+3\lambda-2)-(\lambda+1)\cdot(\lambda^2+\lambda-4)} = \ell^{\lambda^2+\lambda+4},$$

hence we have $\ell \leq n^{\frac{2\lambda+2}{\lambda^2+\lambda+4}}$.

On the other hand, we have

$$\ell^{1/\lambda} \geq \left(\frac{\ell^{\lambda+1}}{n}\right)^{\frac{2}{\lambda^2+3\lambda-2}} \qquad \Longleftrightarrow \qquad n^{\lambda} \leq \ell^{(\lambda+1)\lambda-\frac{\lambda^2+3\lambda-2}{2}} = \ell^{\frac{\lambda^2-\lambda+2}{2}},$$

so the left branch of the $\min(\cdot,\cdot)$ in Eq. 14 is only smaller when $\ell \geq n^{\frac{2\lambda}{\lambda^2-\lambda+2}}$. As

$$\frac{2\lambda+2}{\lambda^2+\lambda+4} \leq \frac{2\lambda}{\lambda^2-\lambda+2},$$

for all $\lambda \geq 1$, it follows that the left branch of the $\min(\cdot,\cdot)$ in Eq. 14 is only smaller than the right when the entire term is not the maximum. Thus

$$x \geq \max_{k\in\mathbb{N}} \left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}.$$

To get a looser but more easily comprehensible lower bound, we note that $\max_{k\in\mathbb{N}} \log\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}$ is piecewise linear and convex in $\log\ell$. Consequently, we can lower bound it using the convex function $C\frac{(\log\ell)^2}{\log n}$, where C is the maximum value which satisfies the inequality at all "corner points" of $\max_{k\in\mathbb{N}} \log\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}$. These corner points occur precisely at values of $\log\ell$ where, for some $k \geq 2$, we have:

$$\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}} = \left(\frac{\ell^{(k-1)+1}}{n}\right)^{\frac{2}{(k-1)^2+3(k-1)-2}}.$$

Rearranging this gives:

$$\frac{\log n}{\log\ell} = \frac{(k+1)((k-1)^2+3(k-1)-2)-(k-1+1)(k^2+3k-2)}{((k-1)^2+3(k-1)-2)-(k^2+3k-2)} = \frac{k^2+k+4}{2k+2}.$$

so the corners occur at $\ell = n^{\frac{2k+2}{k^2+k+4}}$; and at such ℓ, we have

$$\begin{aligned}
\log\left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}} &= \left(\frac{2}{k^2+3k-2}\cdot((k+1)\frac{2k+2}{k^2+k+4}-1)\right)\log n = \frac{2}{k^2+k+4}\log n \\
&= \frac{(\log\ell)^2}{\log n}\frac{2}{k^2+k+4}\left(\frac{\log n}{\log\ell}\right)^2 = \frac{(\log\ell)^2}{\log n}\frac{2}{k^2+k+4}\left(\frac{k^2+k+4}{2k+2}\right)^2 \\
&= \frac{1}{2}\frac{(\log\ell)^2}{\log n}\frac{k^2+k+4}{(k+1)^2} \geq \frac{15}{32}\frac{(\log\ell)^2}{\log n}.
\end{aligned}$$

The function $\frac{k^2+k+4}{(k+1)^2}$ has derivative $\frac{k-7}{(k+1)^3}$ and is minimized exactly at $k=7$, where it has value $\frac{15}{16}$. Consequently, the value $C = \frac{15}{32}$ is the best possible. □

A.3 Mechanical Proofs for Section 5

Proof of Lemma 5.2. First, we handle the case where $\lceil\log\ell\rceil < \left\lfloor 2\frac{\log(n/4)}{\log 16\ell}\right\rfloor$. Then $d = \lceil\log\ell\rceil \leq \left\lfloor 2\frac{\log(n/4)}{\log 16\ell}\right\rfloor - 1$, and $\alpha = 2$. Note that $\frac{\ell}{2^{d-1}} \geq 1$ since $2^{d-1} \leq 2^{\lceil\ell\rceil-1} \leq 2\ell/2 = \ell$.

$$\prod_{i=2}^{d} b_i = \left\lfloor \frac{\ell}{2^{d-1}} \right\rfloor 2^{d-2} \geq \frac{\ell}{2} = \frac{\ell}{\alpha}$$

$$\prod_{i=2}^{d} b_i = \left\lfloor \frac{\ell}{2^{d-1}} \right\rfloor 2^{d-2} \leq 2\frac{\ell}{2} \leq \frac{4\ell}{\alpha}.$$

Since $b_d = \left\lceil \frac{\ell}{2^{d-1}} \right\rceil = \left\lceil \frac{2\ell}{2^{\lceil \log \ell \rceil}} \right\rceil \le 2$,

$$\begin{aligned}
\prod_{i\in[d]} w_i &= (16\ell)\prod_{i=2}^{d}\prod_{j=i}^{d} b_j \le (16\ell)\prod_{i=2}^{d} 2^{d-i+1} = (16\ell)2^{d(d-1)/2} \\
&\le (16\ell)(2^{\lceil \log \ell \rceil})^{(d-1)/2} \le (16\ell)(2\ell)^{(d-1)/2} \\
&\le (16\ell)(2\ell)^{\left(\left\lfloor 2\frac{\log(n/4)}{\log 16\ell}\right\rfloor - 2\right)/2} \\
&\le (16\ell)(2\ell)^{\frac{\log(n/4)}{\log 16\ell}-1} \\
&\le (16\ell)^{\frac{\log(n/4)}{\log 16\ell}} = \frac{n}{4} \le n\,.
\end{aligned}$$

Second, we consider the case where $d = \left\lfloor 2\frac{\log(n/4)}{\log(16\ell)} \right\rfloor$. Because $n \ge 64\ell$, $d \ge 2$, and so

$$d = \left\lfloor 2\frac{\log(n/4)}{\log 16\ell} \right\rfloor \ge \frac{2}{3}\cdot 2\frac{\log(n/4)}{\log 16\ell} = \frac{\log(n/4)}{\frac{3}{4}\log 16\ell} \ge \frac{\log(n/4)}{\log 4\ell}\,.$$

The second inequality used that $\frac{3}{4}(4+\log\ell) \le (2+\log\ell)$ for $\ell \ge 4$. Consequently,

$$\alpha = \left(\frac{(4\ell)^d}{n/4}\right)^{\frac{2}{d(d-1)}} \ge \left(\frac{(4\ell)^{\frac{\log(n/4)}{\log 4\ell}}}{n/4}\right)^{\frac{2}{d(d-1)}} = \left(\frac{n/4}{n/4}\right)^{\frac{2}{d(d-1)}} = 1\,.$$

We now prove Eq. 15. Because $\prod_{i=2}^{d-1} b_i \ge \alpha^{d-2}$,

$$\prod_{i=2}^{d} b_i = \left\lceil \frac{\ell}{\alpha^{d-1}} \right\rceil \prod_{i=2}^{d-1} b_i \ge \frac{\ell}{\alpha \prod_{i=2}^{d-1} b_i} \cdot \prod_{i=2}^{d-1} b_i = \frac{\ell}{\alpha}\,.$$

For Eq. 17, we observe that

$$d \le 2\frac{\log(n/4)}{\log(16\ell)} \qquad \Longrightarrow \qquad 16\ell \le (n/4)^{2/d} \qquad \Longrightarrow \qquad \ell \ge \alpha^{d-1} = \frac{(4\ell)^2}{(n/4)^{2/d}}\,,$$

and thus $\ell/\alpha^{d-1} \ge 1$, so $b_d = \lceil \ell/\alpha^{d-1} \rceil \le 2\ell/\alpha^{d-1}$. Then since $\prod_{i=2}^{d-1} b_i \le 2\alpha^{d-2}$,

$$\prod_{i=2}^{d} b_i \le \frac{2\ell}{\alpha^{d-1}} \prod_{i=2}^{d} b_i \le \frac{4\ell}{\alpha \prod_{i=2}^{d-1} b_i} \cdot \prod_{i=2}^{d-1} b_i \le \frac{4\ell}{\alpha}\,.$$

Finally, we prove Eq. 17. As noted above,

$$b_d \le \frac{2\ell}{\alpha^{d-1}} \le \frac{4\ell}{\alpha \prod_{i=2}^{d-1} b_j}\,.$$

Applying this fact to bound the left hand side of Eq. 17 gives:

$$\begin{aligned}
\prod_{i\in[d]} w_i &= (16\ell)\prod_{i=2}^{d}\prod_{j=i}^{d} b_j = 16\ell(b_d)^d \prod_{i=2}^{d-1}\prod_{j=i}^{d-1} b_j \\
&\le 16\ell\left(\frac{4\ell}{\alpha\prod_{i=2}^{d-1} b_j}\right)^{d-1}\prod_{i=2}^{d-1}\prod_{j=i}^{d-1} b_j \\
&\le \frac{4\cdot(4\ell)^d}{\alpha^{d-1}}\frac{1}{\prod_{i=2}^{d-1}\prod_{j=2}^{i-1} b_j} \\
&\le \frac{4\cdot(4\ell)^d}{\alpha^{d-1}}\frac{1}{\alpha^{(d-1)(d-2)/2}} \qquad \text{since } \prod_{j=2}^{d-1} b_j \ge \alpha^{d-2} \text{ and } b_2 \ge b_3 \ge \ldots \ge b_{d-1} \\
&= \frac{4\cdot(4\ell)^d}{\alpha^{d(d-1)/2}} = \frac{4\cdot(4\ell)^d}{\frac{(4\ell)^d}{n/4}} = n\,.
\end{aligned}$$

□

Proof of Lemma 5.5. Listing 3 only stores two types of data: for each $i \in [d]$, the vectors $L_i \in [w_i]^{b_i}$, and the vectors $x_i \in \{0,1\}^{b_i}$. These can be stored using $b_i \log w_i$ and b_i, bits respectively, for a total of:

$$\begin{aligned}
\sum_{i\in[d]} b_i \log(2w_i) &\le b_1\log(2w_1) + \sum_{i=2}^{d} b_i \log(2w_i) \\
&\le b_1\log(32\ell) + \sum_{i=2}^{d} b_i \log(2\prod_{j=i}^{d} b_i) \le \sum_{i=1}^{d} b_i \log(32\ell)\,,
\end{aligned}$$

since by Eq. 16, $\prod_{j=i}^{d} b_i \le 4\ell$.

We now observe that $b_d \le \lceil\alpha\rceil$. If $d = \lceil\log\ell\rceil$, then $\alpha = b_2 = \ldots = b_{d-1} = 2$ and $b_d = \lceil \ell/\alpha^{d-1}\rceil \le 2$. On the other hand, if $d = \left\lfloor 2\frac{\log(n/4)}{\log(16\ell)}\right\rfloor$, then $\alpha = \frac{(4\ell)^{2/(d-1)}}{(n/4)^{2/(d(d-1))}}$. We have:

$$(4\ell)^{d+1} = (4\ell)^{\left\lfloor 2\frac{\log(n/4)}{\log(16\ell)}\right\rfloor+1} \ge (4\ell)^{2\frac{\log(n/4)}{\log(16\ell)}} = (n/4)^2\,,$$

which implies

$$\alpha = \frac{(4\ell)^{2/(d-1)}}{(n/4)^{2/(d(d-1))}} \ge \frac{(4\ell)^{2/(d-1)}}{(4\ell)^{(d+1)/(d(d-1))}} = (4\ell)^{(2-\frac{d+1}{d})\cdot\frac{1}{d-1}} = (4\ell)^{\frac{1}{d}}\,.$$

Consequently,

$$b_d = \left\lceil \frac{\ell}{\alpha^{d-1}}\right\rceil = \left\lceil \frac{1}{4}\frac{4\ell}{\alpha^{d-1}}\right\rceil \le \left\lceil \frac{1}{4}(4\ell)^{1/d}\right\rceil \le (4\ell)^{1/d} \le \alpha\,.$$

With the bound on b_d, and the fact that $\alpha \ge 1$ in both cases, and that $d \le \lceil\log\ell\rceil$ we obtain:

$$\begin{aligned}
\sum_{i\in[d]} b_i &\le \min(\ell+1, \lceil 8\alpha\rceil + \lceil 3\log 1/\delta\rceil) + (d-1)\lceil\alpha\rceil \\
&\le \min(\ell, \lceil 3\log 1/\delta\rceil) + (7+d)2\alpha \\
&\le \min(\ell, \lceil 3\log 1/\delta\rceil) + 32\log\ell\left\lceil \frac{(4\ell)^{2/(d-1)}}{(n/4)^{2/(d(d-1))}}\right\rceil.
\end{aligned}$$

Multiplying this last quantity by $\log(32\ell)$ gives a space bound.

To obtain a much weaker, but somewhat more comprehensible upper bound on α, when $d = \left\lfloor 2\frac{\log(n/4)}{\log(16\ell)} \right\rfloor$, we note that:

$$\begin{aligned}\max_{\lambda\in\mathbb{N}\cap[2,\infty)} \log \frac{(4\ell)^{2/(\lambda-1)}}{(n/4)^{2/(\lambda(\lambda-1))}} &\leq \log \max_{\lambda\in\mathbb{R}\cap[2,\infty)} \left(\frac{2}{\lambda-1}\log(4\ell) - \frac{2}{\lambda(\lambda-1)}\log(n/4)\right) \\ &\leq \log\left(2\log(4\ell) \max_{\lambda\in\mathbb{R}\cap[2,\infty)} \left(\frac{1}{\lambda-1} - \frac{1}{\lambda(\lambda-1)}\frac{\log(n/4)}{\log(4\ell)}\right)\right).\end{aligned}$$

Let $\gamma = \frac{\log(n/4)}{\log(4\ell)}$; this is ≥ 1. Let $f(x) = \frac{1}{x-1}(1-\frac{\gamma}{x})$. We will now prove that $\max_{x\geq 2} f(x) \leq \frac{1}{2\gamma}$. We note that when $x = 2$, we have:

$$f(2) = 1 - \frac{\gamma}{2} \leq \frac{1}{2\gamma}.$$

Checking the other endpoint, we have:

$$\lim_{x\to\infty} \frac{1}{x-1}(1-\frac{\gamma}{x}) = 0.$$

Since $f(x)$ is differentiable on $[2,\infty)$, if it has a maximum other than at the endpoints, then it will occur when $\frac{d}{dx}f(x) = 0$. Solving this equation, we obtain:

$$\frac{d}{dx}f(x) = -\frac{1}{(x-1)^2} + \frac{\gamma(2x-1)}{(x(x-1))^2} = -\frac{1}{(x-1)^2}\left[1 - \frac{\gamma(2x-1)}{x^2}\right] = 0,$$

which is true iff $x^2 = \gamma(2x-1)$. The solutions to the quadratic equation are

$$x = \gamma - \sqrt{\gamma(\gamma-1)} \qquad \text{and} \qquad x = \gamma + \sqrt{\gamma(\gamma-1)}.$$

Since $\gamma \geq 1$, the $-$ branch has $x \leq 1$, which is not in $[2,\infty)$. The $+$ branch is only in $[2,\infty)$ if $\gamma \geq \frac{4}{3}$. The value of $f(x)$ in this case is:

$$\begin{aligned} f(\gamma+\sqrt{\gamma(\gamma-1)}) &= \frac{1}{\gamma+\sqrt{\gamma(\gamma-1)}-1}\left(1 - \frac{\gamma}{\gamma+\sqrt{\gamma(\gamma-1)}}\right) \\ &= \frac{\sqrt{\gamma(\gamma-1)}}{2\gamma-1+2\sqrt{\gamma(\gamma-1)}} \\ &\leq \frac{1}{4\sqrt{\gamma(\gamma-1)}} && \text{(since } \sqrt{\gamma(\gamma-1)} \leq 2\gamma-1 \text{ for all } \gamma \geq 1\text{)} \\ &\leq \frac{1}{2\gamma}. && \text{(since } \gamma \leq 2\sqrt{\gamma(\gamma-1)} \text{ for all } \gamma \geq \frac{4}{3}\text{)} \end{aligned}$$

Thus, if $f(x)$ does have a maximum in $[2,\infty)$, it is $\leq \frac{1}{2\gamma}$. We conclude that $f(x) \leq \frac{1}{2\gamma}$ in all cases. This proves:

$$\log\alpha \leq \max_{\lambda\in\mathbb{N}} \log \frac{(4\ell)^{2/(\lambda-1)}}{(n/4)^{2/(\lambda(\lambda-1))}} \leq \log\left(2\log(4\ell)\frac{1}{2^{\frac{\log(n/4)}{\log(4\ell)}}}\right) \leq \log\frac{\log(4\ell)^2}{\log(n/4)}. \qquad \square$$

A.4 Mechanical Proofs for Section 6

Proof of Theorem 6.9. By Lemma 6.7, we have:

$$z \geq \min\left(\frac{\ell}{36\log\frac{2|S|}{\ell}}, \frac{\ell}{17280\log\frac{2|S|}{\ell}} \cdot \frac{\log(1/2\delta)}{\log(64|S|)\log\frac{2|S|}{\ell}}\right). \tag{30}$$

By Lemma 6.8, $|S| \leq \min(n, 2^{z+1}) \leq \min(n, 4^z)$. We will apply this inequality to each branch of the minimum in Eq. 30. First, say that the left part of the minimum is larger than the right. Then $z \geq \ell/(36\log\frac{2|S|}{\ell})$. Applying $|S| \leq n$ and $|S| \leq 4^z$, this implies:

$$z \geq \frac{\ell}{36\log\frac{2n}{\ell}}, \qquad \text{and}$$
$$z \geq \frac{\ell}{36\log\frac{2\cdot 4^z}{\ell}} \geq \frac{\ell}{36\cdot 3z} \qquad \Longrightarrow \qquad z \geq \sqrt{\frac{\ell}{108}}.$$

Thus:

$$z \geq \max\left(\frac{\ell}{36\log\frac{2n}{\ell}}, \sqrt{\frac{\ell}{108}}\right). \tag{31}$$

Next, say that the right side of the minimum in Eq. 30 is larger. Then applying $|S| \leq n$ and $|S| \leq 4^z$ to that side, we get:

$$z \geq \frac{\ell\log(1/2\delta)}{17280(\log\frac{2n}{\ell})^2\log(64n)}, \qquad \text{and}$$
$$z \geq \frac{\ell\log(1/2\delta)}{17280(\log\frac{2\cdot 4^z}{\ell})^2\log(64\cdot 4^z)} \geq \frac{\ell\log(1/2\delta)}{17280\cdot 3^2\cdot 8z^3} \qquad \Longrightarrow \qquad z \geq \left(\frac{\ell\log(1/2\delta)}{1244160}\right)^{1/4}.$$

Thus:

$$z \geq \max\left(\frac{\ell\log(1/2\delta)}{17280(\log\frac{2n}{\ell})^2\log(64n)}, \left(\frac{\ell\log(1/2\delta)}{1244160}\right)^{1/4}\right). \tag{32}$$

The minimum of the lower bounds from Eqs. 31 and 32 holds in all cases, so:

$$z \geq \min\left(\max\left(\frac{\ell}{36\log\frac{2n}{\ell}}, \sqrt{\frac{\ell}{108}}\right), \max\left(\frac{\ell\log(1/2\delta)}{17280(\log\frac{2n}{\ell})^2\log(64n)}, \left(\frac{\ell\log(1/2\delta)}{1244160}\right)^{1/4}\right)\right).$$

□

Proof of Theorem 6.11. The lower bound from Theorem 6.9 for $\text{MIF}(n,t)$ with error $\delta = 1/3$, showing constants, is:

$$\max\left(\frac{t\log(3/2)}{17280(\log\frac{2n}{t})^2\log(64n)}, \left(\frac{t\log(3/2)}{1244160}\right)^{1/4}\right). \tag{33}$$

If the space used by an algorithm, z, satisfies $z \geq (\ell-1)/2$, then we tautologically have a lower bound of $(\ell-1)/2$. Otherwise, we have $2z+2 \leq \ell$. Applying Theorem 6.10 gives, for the left branch of the max in Eq. 33, with $t = \lfloor\frac{\ell}{2z+2}\rfloor \geq \frac{1}{8z}$:

$$z \geq \left\lfloor\frac{\ell}{2z+2}\right\rfloor \frac{\log(3/2)}{17280(\log\frac{2n}{t})^2\log(64n)} \geq \frac{\ell}{8z}\frac{1/2}{17280(\log(2n))^2\log(64n)},$$

which implies

$$z \geq \sqrt{\frac{\ell}{276480(\log(2n))^2 \log(64n)}} \geq \sqrt{\frac{\ell}{7741440(\log n)^3}}.$$

For the right branch of Eq. 33, we obtain:

$$z \geq \left(\left\lfloor \frac{\ell}{2z+2} \right\rfloor \frac{\log(3/2)}{1244160} \right)^{1/4} \geq \left(\frac{\ell}{8z} \frac{1}{2 \cdot 1244160} \right)^{1/4},$$

which implies:

$$z^{5/4} \geq \left(\frac{\ell}{19906560} \right)^{1/4} \qquad \Longrightarrow \qquad z \geq \left(\frac{\ell}{19906560} \right)^{1/5}.$$

Combining the two lower bounds, gives:

$$z \geq \max \left(\sqrt{\frac{\ell}{7741440(\log n)^3}}, \left(\frac{\ell}{19906560} \right)^{1/5} \right) = \Omega \left(\sqrt{\frac{\ell}{(\log n)^3}} + \ell^{1/5} \right). \tag{34}$$

This lower bound is everywhere smaller than $(\ell-1)/2$, so it is compatible with the case in which $z \geq (\ell-1)/2$.

Taking the maximum of Eq. 34 and the known random oracle lower bound for $\text{MIF}(n,\ell)$ algorithms in the static setting, Theorem 3.5, gives:

$$z = \Omega \left(\frac{\ell^2}{n} + \sqrt{\frac{\ell}{(\log n)^3}} + \ell^{1/5} \right). \qquad \square$$