Testing forbidden order-pattern properties on hypergrids

Harish Chandramouleeswaran* Ilan Newman† Tomer Pelleg Nithin Varma‡

Abstract

Given a permutation $\pi\colon [k] \to [k]$, a function $f\colon [n]^d \to \mathbb{R}$ is said to be π-free if there are no k indices $x_1 \prec \cdots \prec x_k \in [n]^d$ such that $f(x_i) < f(x_j)$ and $\pi(i) < \pi(j)$ for all $i, j \in [k]$, where $\prec$ is the natural partial order over $[n]^d$. For a fixed π and $\epsilon \in (0,1)$, the problem of ϵ-testing π-freeness is to distinguish the case that f is π-free from the case that at least ϵn^d values of f need to be modified in order to make it π-free. When $k = 2$, the problem is identical to monotonicity testing, which is extensively studied in property testing. The case of $k > 2$ has also received significant attention for functions $f\colon [n] \to \mathbb{R}$.

We initiate a systematic study of testing of π-freeness for higher dimensional grids; specifically, for permutations of size 3 over hypergrids of dimension 2. We design an adaptive one-sided error π-freeness tester with query complexity $O(n^{4/5+o(1)})$ that works for all permutations of size 3. We then argue that for general permutations of size 3, every nonadaptive tester must have query complexity $\Omega(n)$, and further, that every *adaptive* tester must have query complexity $\Omega(\sqrt{n})$. This is the first general lower bound for testing π-freeness that is higher than $\Theta(\log n)$.

For the important special case of $\pi = (1,2,3)$ or $(3,2,1)$, we design a nonadaptive tester with a significantly improved (nearly optimal) query complexity of $\operatorname{polylog} n$. Such an exponential gap in the complexity of testing freeness of nonmonotone and monotone patterns is not known, in general, in the one-dimensional case.

Erasure-resilient (ER) monotonicity testers are important subroutines in the design of our π-freeness testers. We design δ-ER ϵ-testers for monotonicity for functions $f\colon [n]^d \to \mathbb{R}$ with query complexity $O\left(\frac{\log^{O(d)} n}{\epsilon(1-\delta)}\right)$, where $\delta \in (0,1)$ is an upper bound on the fraction of erasures. Earlier erasure-resilient monotonicity testers worked only when $\delta = O(\epsilon/d)$. The complexity of our nonadaptive monotonicity tester is nearly optimal as evidenced by a lower bound of Pallavoor, Raskhodnikova, and Waingarten (Random Struct. Algorithms, 2022).

Lastly, we argue that the current techniques in function property testing cannot give us sublinear-query testers for patterns of length 4 even for 2-dimensional hypergrids.

1 Introduction

Order relations between values capture both local and global structural properties of numerical datasets. The study of specific order patterns in inputs is a fundamental topic spanning algorithms, combinatorics and applications. Knuth, in an early work [Knu68], showed that arrays that avoid the pattern $(2,3,1)$ are exactly the ones that can be sorted by a single stack. Recent research in the same spirit has focused on designing faster algorithms for optimization problems such as Euclidean

*Chennai Mathematical Institute, India (harishc@cmi.ac.in).

†Department of Computer Science, University of Haifa, Israel (ilan@cs.haifa.ac.il). Research supported by the Israel Science Foundation (grant 379/21).

‡Department of Mathematics and Computer Science, University of Cologne, Germany (varma@cs.uni-koeln.de).

TSP and searching in BSTs when inputs avoid certain order patterns [CGK+15, CGJ+23, CPY24, BKO24].

There is also a long line of work in combinatorics on the number of permutations that avoid specific order patterns [Bón97, Bón99, Arr99, Kla00, AF00], culminating in the celebrated Marcus-Tardos theorem [MT04]. There are measures based on the frequencies of various order patterns such as permutation entropy[1], which are extensively used in time series analysis [BP02] and have applications to varied fields such as biomedical sciences and econophysics among others [ZZRP12, Ban16].

Given a real-valued function $f\colon [n] \to \mathbb{R}$ and a permutation (order pattern) $\pi\colon [k] \to [k]$, we say that f contains a π-*appearance* if there exist distinct indices $x_1, \ldots, x_k \in [n]$ such that the values of f respect the ordering dictated by π. That is, for all $i, j \in [k]$, $f(x_i) < f(x_j)$ if and only if $\pi(i) < \pi(j)$. The function f is said to be π-*free* otherwise. There are several classical algorithms for deciding whether a given function is π-free for short patterns π [AAAH01, AR08, BKM21], including linear-time algorithms [GM14, Fox13]. The quest for faster algorithms has led to the study of π-freeness in the property testing framework [RS96, GGR98] of sublinear-time algorithms, although much less is known in this setting. Investigations on forbidden order pattern testing has so far focused mostly on the case of real-valued functions on the line $[n]$. Formally, for a parameter $\epsilon \in (0, 1)$, a function $f\colon [n] \to \mathbb{R}$ is ϵ-far from being π-free if at least ϵn values of f must be modified to make it π-free. An ϵ-tester for π-freeness is an algorithm that, given oracle access to f, decides with constant probability whether f is π-free or ϵ-far from π-free. For general constant length patterns and constant ϵ, the best known π-freeness testers have query complexity $\widetilde{O}(n^{o(1)})$ [NV24]. In contrast, for the important special case of monotonicity testing (i.e., $(2,1)$-freeness testing), optimal ϵ-testers [EKK+00, Fis04, BGJ+12, CS13, Bel18] have query complexity $\Theta\left(\frac{\log n}{\epsilon}\right)$. Research on testing order pattern freeness has also resulted in the development of several interesting combinatorial techniques [NRRS19, BCLW19, BC18, BLW22, NV24].

The significance of the study of short ordered patterns in one-dimensional functions is evident from the discussion above. Many of the aforementioned problems extend naturally to higher-dimensional functions and are motivated by both theoretical interest and practical applications, as, for instance, in analyzing higher dimensional time series data [KL03, RZL+12]. In the context of sublinear-time algorithms, monotonicity testing has been extensively studied for functions over general hypergrids, i.e., for $f\colon [n]^d \to \mathbb{R}$, and there are optimal ϵ-testers with complexity $\Theta\left(\frac{d \log n}{\epsilon}\right)$ [GGL+00, DGL+99, CS13, CS14, CDJS17]. However, little is known for testing general pattern freeness over domains other than the line, and in particular, over hypergrids of dimension larger than 1.

1.1 Our results

In this paper, we address this knowledge gap by initiating a systematic study of pattern freeness testing of real-valued functions over higher dimensional hypergrid domains. Let $\prec$ denote the natural partial order over $[n]^d$, i.e., $x \prec y$ if, for all $i \in [d]$, we have $x[i] \leq y[i]$, where $p[i]$ refers to the i^{th} coordinate of $p \in [n]^d$. Given a permutation $\pi \in \mathcal{S}_k$, a function $f\colon [n]^d \to \mathbb{R}$ is π-free if there are no

[1]The presence or absence of specific order patterns is used to empirically determine whether a time series is stochastic or deterministic in nature. Specifically, the time series is inferred to be deterministic if it avoids some specific order pattern [RLM+07, ZSR12] There are also dedicated libraries to perform various ordinal analyses on sequential datasets [BKSJ19, PR21].

k indices $x_1 \prec \cdots \prec x_k \in [n]^d$ such that $f(x_i) < f(x_j)$ and $\pi(i) < \pi(j)$ for all $i, j \in [k]$. For a fixed π and $\epsilon \in (0,1)$, the problem of ϵ-testing π-freeness is to distinguish the case that f is π-free from the case that at least ϵn^d values of f need to be modified in order to make it π-free. The problem is interesting in its own right and as a natural generalization of monotonicity testing for functions over the hypergrids.

Our results can be summarized as follows. We develop techniques to design sublinear algorithms and strong lower bounds for testing π-freeness over hypergrids for all patterns $\pi \in \mathcal{S}_3$. It turns out that there are interesting new phenomena and significant new challenges to deal with for patterns of length 3 even over 2-dimensional hypergrids. For testing freeness of larger order patterns, we argue that the current techniques in function property testing cannot give strong sublinear-query testers even over 2-dimensional hypergrids. Erasure-resilient as well as partial function monotonicity testing are important components of our algorithms. Naturally, we design nearly optimal fully erasure-resilient monotonicity testers [DRTV18] over the hypergrids and efficient monotonicity testers over high dimensional partial functions.

Order-patterns over hypergrids

We show that there are fundamental distinctions between nonmonotone and monotone patterns as well as between large and small patterns for π-freeness testing over the hypergrids. We emphasize that this is very much unlike the case for the line $[n]$. We also show a distinction between adaptive and nonadaptive algorithms for π-freeness over hypergrids. The bounds we obtain are summarized in Table 1.

Monotone vs. nonmonotone patterns. Much of the research on π-freeness of functions $f\colon [n] \to \mathbb{R}$ has focused on the case of π being monotone [NRRS19, BC18, BCLW19, BLW22]. For constant k and ϵ, Ben-Eliezer, Letzter, and Waingarten [BLW22] designed ϵ-testers for π-freeness with query complexity $O(\log n)$ for monotone patterns π of length k, i.e., for $\pi \in \{(1,2,\ldots,k), (k,\ldots,2,1)\}$. The best-known ϵ-tester for general π-freeness, due to Newman and Varma [NV24], has query complexity $\widetilde{O}(n^{o(1)})$ for constant k and ϵ. Additionally, the only known general lower bound for ϵ-testing freeness of patterns of *any* length is $\Omega(\log n)$, which is the lower bound for monotonicity testing [Fis04]. Newman and Varma [NV24] conjecture that one can design π-freeness testers for all patterns π of constant length with query complexity $\operatorname{polylog} n$. They base their conjecture mainly on the fact that one can test π-freeness for all patterns π of length 3 with query complexity $\operatorname{polylog} n$ [NRRS19].

In this paper, we show that an analogous conjecture cannot hold for pattern-freeness testing over general hypergrids. Specifically, we prove that testing nonmonotone pattern freeness is exponentially harder than testing monotone pattern freeness *even for patterns of length* 3 *over* 2*-dimensional hypergrids.*

Our first main result is an $\Omega(\sqrt{n})$ lower bound for π-freeness testing of patterns π of length 3 of functions $f\colon [n]^2 \to \mathbb{R}$. We emphasize that our lower bound holds even when the tester is allowed to make its queries adaptively.[2]

Theorem 1.1. *Any one-sided error*[3] *ϵ-tester for $(1,3,2)$-freeness of functions $f : [n]^2 \to \mathbb{R}$, has query complexity $\Omega(\sqrt{n})$, for every $\epsilon \le 4/81$.*

[2] A tester is *nonadaptive* if its queries do not depend on the answers to previous queries. It is *adaptive* otherwise.

[3] A tester has *one-sided error* if it accepts a π-free function with probability 1. It has *two-sided error* otherwise.

The above is the first nontrivial adaptive lower bound for pattern freeness that does not directly follow from the lower bounds for monotonicity testing [Fis04, CS14]. We mention that our techniques can be generalized to higher-dimensional functions $f\colon [n]^d \to \mathbb{R}$ to show an $\Omega(n^{(d-1)/2})$ lower bound for general π-freeness of patterns π of length 3. The lower bounds that we obtain are exponentially stronger than the $\Omega(d \log n)$ bounds that directly comes out of monotonicity testing over $f\colon [n]^d \to \mathbb{R}$ [Fis04, CS14].

In contrast, for monotone 3-patterns, i.e., when $\pi = (1, 2, 3)$, or, equivalently, $(3, 2, 1)$, we design testers with exponentially better query complexity. Our testers have the additional feature of being nonadaptive. In particular, our result can be seen as a higher dimensional counterpart of the fact that testing monotone patterns of constant length can be done *nonadaptively* using $\operatorname{polylog} n$ queries for real-valued functions over the line $[n]$ [NRRS19, BCLW19]. However, proving our result requires very different ideas.

Theorem 1.2. *There exists a one-sided error nonadaptive ϵ-tester with query complexity $\log^{O(1)} n$ for $(1, 2, 3)$-freeness of functions $f\colon [n]^2 \to \mathbb{R}$ for constant $\epsilon \in (0, 1)$.*

We remark that such a gap between testing nonmonotone and monotone pattern freeness is known for the line $[n]$ when restricted to *nonadaptive testers*. Ben-Eliezer and Canonne [BC18] showed that there are nonmonotone patterns π of length k such that every nonadaptive π-freeness tester requires $\Omega(n^{1-(1/(k-1))})$ queries. However, there are nonadaptive testers for all monotone patterns of length k that has query complexity $\Theta((\log n)^{\log k})$ [BCLW19].

Adaptivity vs. nonadaptivity. We also complement our lower bound in Theorem 1.1 with efficient sublinear-time π-freeness testers for general patterns π of length 3. Using standard birthday paradox arguments, a simple ϵ-tester with query complexity $O(n^{4/3}/\epsilon^{1/3})$ for testing π-freeness of patterns π of length 3 of functions $f\colon [n]^2 \to \mathbb{R}$ (see Theorem 2.7) can be obtained. This is already a sublinear-query tester as the size of the domain is n^2.[4] Furthermore, the tester has one-sided error and is nonadaptive. However, this tester is only *moderately* sublinear, and in particular, its query complexity is larger than n.

We show that, in general, nonadaptive testers for π-freeness of patterns π of length 3 of functions $f\colon [n]^2 \to \mathbb{R}$ cannot do significantly better.

Theorem 1.3. *Any one-sided error nonadaptive ϵ-tester for π-freeness of functions $f : [n]^2 \to \mathbb{R}$, for general $\pi \in \mathcal{S}_3$, has query complexity $\Omega(n)$.*

Our lower bound technique can also yield a query complexity lower bound of $\Omega(n^{d-1})$ on the nonadaptive query complexity of testing π-freeness of functions $f\colon [n]^d \to \mathbb{R}$ for general $\pi \in \mathcal{S}_3$.

We also design $o(n)$-query adaptive testers for the same problem proving that adaptivity can and does significantly help π-freeness testing.

Theorem 1.4. *Let $\pi \in \mathcal{S}_3$. There exists a one-sided error ϵ-tester with query complexity $O(n^{4/5+o(1)})$ for π-freeness of functions $f\colon [n]^2 \to \mathbb{R}$ for constant $\epsilon \in (0, 1)$.*

Our result is in contrast to the case of monotonicity, where adaptivity is known to not help, in general, even over hypergrid domains [CS14]. Moreover, these are the first nontrivial sublinear-query π-freeness testers for functions $f\colon [n]^d \to \mathbb{R}$ and $\pi \in \mathcal{S}_3$ for $d > 1$.

[4] This argument is more general and applies to longer patterns and higher dimensions as long as some additional requirements on the (Hamming) distance to pattern freeness are met. See Section 2.1 for more details.

Large vs. small order patterns. For patterns of length 4 or more, we provide evidence that the current techniques cannot give us sublinear-query testers for dimension larger than 1. The reason is that all known π-freeness testers exploit the fact that functions that are far from being π-free have many disjoint π-appearances. We show in Section 2.1 that such a statement is not true in general for patterns of length 4 or more over hypergrids. This is in stark contrast to the case of functions defined on the line, where this fact is crucially used to obtain sublinear-query testers for all constant-length patterns [NV24].

Erasure-resilient monotonicity testing

The *erasure-resilient model* (henceforth referred to as the ER model), introduced by Dixit, Raskhodnikova, Thakurta, and Varma [DRTV18], is the setting where some of the function values are "erased" by an adversary and cannot be used by the algorithm. An ER algorithm gets an upper bound $\delta \in (0,1)$ on the fraction of erased values. The actual erased points are unknown and the algorithm learns whether the value at a specific domain point has been erased by the adversary only when that point is queried. Given parameters $\delta \in (0,1)$ and $\epsilon \in (0,1)$, a δ-ER ϵ-tester for a property $\mathcal{P}$ of functions $\{f\colon D \to \mathbb{R}\}$ gets oracle access to a function f with at most δ-fraction of adversarially erased values in unknown places. The tester must decide whether f can be completed to a function in $\mathcal{P}$ or whether every completion of f must be changed in at least an ϵ-fraction of the nonerased domain in order to belong to $\mathcal{P}$.[5]

Erasure-resilient monotonicity testers are crucial components of our pattern freeness testers. Dixit et al. [DRTV18] designed optimal δ-ER ϵ-testers for monotonicity of functions $f\colon [n]^d \to \mathbb{R}$ with complexity $O\left(\frac{d \log n}{\epsilon(1-\delta)}\right)$, *but* these testers only work when the fraction of erasures δ is $O(\epsilon/d)$. However, Pallavoor, Raskhodnikova, and Waingarten [PRW22] proved that every nonadaptive δ-ER ϵ-tester for monotonicity of real-valued functions $f\colon \{0,1\}^d \to \mathbb{R}$ on the Boolean hypercube must have query complexity 2^{d^κ} even when $\delta = \Theta\left(\frac{\epsilon}{d^{0.5-\kappa}}\right)$, where $\kappa \in (0, 0.5)$. We design optimal ER-testers for functions $f\colon [n]^d \to \mathbb{R}$ that work for any fraction of erasures.

Theorem 1.5. *There exists a δ-ER ϵ-tester for monotonicity of functions $f\colon [n]^d \to \mathbb{R}$ with query complexity $O\left(\frac{\log^{O(d)} n}{\epsilon(1-\delta)} \cdot \log^2 \frac{1}{\epsilon}\right)$, for any $0 < \delta, \epsilon < 1$.*

Monotonicity testing of partial functions

Testing properties of partial functions is a problem closely related to erasure-resilient testing. A *partial* function is a function $f\colon P \to \mathbb{R}$, for a given (and known) subset $P \subseteq [n]^d$. For the line, testing pattern freeness of partial functions is identical to testing total functions (for any pattern), as the grid order induced on P is a total order (that is, identical to the line order). However, for higher-dimensional grids ($d \geq 2$), the order induced on P is not identical to the grid order. In particular, while any tester for total functions on the line can be applied to test properties of partial functions with the same complexity (w.r.t. the domain size $|P|$), this is no longer true for hypergrids. We design monotonicity testers for partial functions on the d-dimensional grid with query complexity $O(\log^{O(d)} n)$, which is independent of the size of P. We emphasize that this result does not directly follow by treating the points in $[n]^d \setminus P$ as erased, and then using the erasure-resilient tester as a blackbox.

[5] A completion of a partially erased function is a total function agreeing with it on the nonerased points.

Table 1: Summary of the relevant results in pattern freeness testing

Pattern length	Pattern type	$d=1$ (Line)		$d=2$ (Hypergrid)	
		Nonadaptive	Adaptive	Nonadaptive	Adaptive
$k=2$	$\Theta(\log n)$ [EKK+00, Fis04, CS13, CS14]				
$k=3$	Monotone patterns	$\Theta(\log n)$ [BCLW19]	$\Theta(\log n)$ [BLW22]	polylog(n) (Theorem 1.2)	?
	Nonmonotone patterns	$\Theta(\sqrt{n})$ [BC18, NRRS19]	polylog(n) [NRRS19]	$O(n^{4/3})$ (Theorem 2.7), $\Omega(n)$ (Theorem 1.3)	$O(n^{4/5+o(1)})$ (Theorem 1.4), $\Omega(\sqrt{n})$ (Theorem 1.1)

1.2 Related work

The study of testing forbidden order-patterns on real-valued arrays (i.e., real-valued functions on the line) was initiated by Newman, Rabinovich, Rajendraprasad, and Sohler [NRRS19]. For monotone patterns π of length k, they designed a nonadaptive one-sided error ϵ-tester for π-freeness with query complexity $(\epsilon^{-1}\log n)^{O(k^2)}$. For nonmonotone patterns, they gave an $\Omega\left(n^{1-\frac{2}{k+1}}\right)$ lower bound and an $O\left(n^{1-\frac{1}{k}}\right)$ upper bound in the nonadaptive setting, and an adaptive polylog-query tester for 3-patterns. Ben-Eliezer and Canonne [BC18] improved the bounds for nonadaptive testing of nonmonotone k-patterns to the essentially tight $\Theta\left(n^{1-\frac{1}{k-1}}\right)$. Subsequently, the complexity of testing monotone patterns was improved in a sequence of works [BCLW19, BLW22], to $O_{k,\epsilon}(\log n)$, which is optimal for constant ϵ even for the special case of testing $(2,1)$-freeness [Fis04]. For general k-patterns π, Newman and Varma [NV24] designed a $\widetilde{O}\left(n^{o(1)}\right)$-query adaptive one-sided error ϵ-tester for π-freeness. ER-testing of general patterns on the line has not been considered per se, but the results in [NV24] can be made ER.

Yet another related direction is pattern freeness testing of functions over $[n]^d$ *with a constant-sized range*. Fischer and Newman [FN07] designed a one-sided error ϵ-tester with query complexity that depends only on d and ϵ, for arbitrary (including non-permutation) patterns of constant length. We mention that the dependence of the query complexity on d and ϵ can even be doubly exponential in some cases.

Lastly, we note that there is also a large amount of literature devoted to studying monotonicity ($(2,1)$-freeness) of Boolean functions and real-valued functions over high-dimensional domains [GGL+00, DGL+99, FLN+02, CS16, CST14, CDST15, CWX17, KMS18, BCS18, PRV18, BCS20, BKR23, BKKM23, BCS23b, BCS23a], but these are less relevant to the current work.

1.3 Technical ideas and challenges

All testing algorithms for π-freeness where π is a permutation pattern, i.e., $\pi \in \mathcal{S}_k$ where $\mathcal{S}_k$ is the symmetric group on k elements (including the restricted case of $\pi \in \mathcal{S}_2$ (i.e., monotonicity)) are crucially based on two ingredients.

The first ingredient is a relation between two different distances, the *Hamming distance* and the *deletion distance* (Theorem 2.1). All testers in the literature are designed to distinguish between functions that are π-free (i.e., satisfy the property), and functions that are far from being π-free with respect to the *deletion distance*. However, the standard definition of testing is with respect to the Hamming distance. Fortunately, for the properties of π-freeness on the line, and for monotonicity (in any dimension), these two distances are equal. We show that for $d \geq 2$, this is not the case in general, even for some 4-patterns. However, for 3- and 2-patterns, this equivalence still holds. This is partially the reason why our results are limited to patterns of size 2 and 3.

The second ingredient for testing larger patterns is to perform some sort of reduction to testing smaller patterns on partial domains, i.e., testing properties of partial functions. This is done extensively in [NRRS19, BCLW19, BLW22, NV24]. For a subdomain $P \subseteq [n]$ of the line, the order induced on P is a total order isomorphic to the order on a (smaller) line. Thus, when the domain is a line, testing partial functions is the same as testing total functions. However, for $P \subseteq [n]^d,\ d \geq 2$, the order on P *is not*, in general, isomorphic to a grid order, making the reduction more involved. Testing of partial functions is a different problem in nature, which is of independent interest. When the density of the subdomain P (i.e., $|P|/n^d$) is a large constant, it may be regarded as an instance of ER-testing. However, we use such reductions extensively for low-density sets. Hence, we were required to develop an ER-monotonicity tester for partial functions, and for (some special) partial functions for 3-patterns. This, in turn, introduces some new challenges and techniques. We now describe these in more detail for the 2-dimensional case, i.e., for functions $f : [n]^2 \to \mathbb{R}$.

Monotonicity testing. For testing monotonicity of partial functions, we build on the idea of [NRRS19] which is referred to as "median-split and concatenate" (or the *median argument* for short). The idea for the line is as follows: suppose that a function f is ϵ-far from $(1,2)$-freeness (that is, far from being monotone decreasing). The median argument consists of sampling two disjoint intervals L and R on the line, where L is on the 'left' of R, and such that: (1) f contains a large set of pairwise disjoint $(1,2)$-appearances $A = \{(p_1, p_2) \in L \times R,\ f(p_1) < f(p_2)\}$, and (2) one can sample a point x such that for a large number of appearances (p_1, p_2) in A, $f(p_1) < f(x)$, and $f(x) < f(p_2)$. Then, such a sampled point x can be used to find a $(1,2)$-appearance of the type described above as follows: sample a point $p \in L$, and a point $q \in R$, uniformly and independently at random. By (1) and (2), with high probability, p and q are the left and right coordinates of some (not necessarily the same) $(1,2)$-appearance in A, where $f(p) < f(x) < f(q)$. This is how a $(1,2)$-appearance (p,q) is found.

Directly applying this argument for monotonicity on $[n]^d,\ d \geq 2$ raises two difficulties. Firstly, the probability of success (even for the line) is proportional to ϵ^2, which will result in a nonoptimal dependence on the distance even for the 1-dimensional case. Secondly, for partial functions and in the ER setting, we must make sure that the value of the function $f(x)$ for the 'median point' x described above is defined and not erased. We further note that for the argument above, one does not need to know x, but merely assure its existence (w.h.p.). We show that the median argument can be generalized to subgrids of $[n]^d$, which will play the roles of the intervals L and R above. This can be done even for partial functions (regardless of the density $|P|/n^d$), and in the presence of any

fraction of erasures (i.e., an ER-test). To be able to use this in our tests for 3-patterns, we require, and obtain, some extra guarantee from the 'median point' x – namely, that it is somewhat 'typical' (see Theorem 3.3).

Testing $(1,2,3)$-freeness. Testing $(1,2,3)$-freeness (and larger monotone patterns) on the line was done in [NRRS19] by extending the median argument as follows. It is shown that for two suitably chosen intervals L and R as above, there are many $(1,2,3)$-appearances that induce $(1,2)$-appearances in L, and a corresponding 3-appearance in R. Then, by the median argument, one can find a suitable $(1,2)$-appearance in L, and complete it to a $(1,2,3)$-appearance in $L \cup R$ by sampling a point in R. However, this is true only for specific ranges of distances between points forming $(1,2,3)$-appearances[6] and cannot be directly generalized to $[n]^2$.

Instead, we partition $[n]^2$ into m^2 blocks of $\left(\frac{n}{m}\right)^2$ size for a certain parameter m. Now, the $(1,2,3)$-appearances in f can be partitioned into a constant number of configurations, according to the relative order of the blocks in which the corresponding 1-, 2-, and 3-legs fall. See e.g., Figure 2. For some of these configurations, we can directly apply the median argument and obtain a tester. For the others, a more sophisticated argument is used to further partition a corresponding region and to finally reduce the problem to monotonicity testing. This is done by extending the median argument to not only the corresponding values of the points as described above, *but also to the coordinates of the points*. These cannot be done simultaneously but can be achieved using the 'correct' order of sampling points, and the 'typical nature of the median point', as mentioned earlier. This is done, for instance, in Case 4.2.4 of the algorithm given in Section 4.2. Unlike in the 1-dimensional case, two additional features to be taken care of are: (1) the reduction is to monotonicity testing with subconstant distances, and (2) the reduced problem is on a partial domain which is not isomorphic to a grid. Using, on top of this, a delicate recursion for an appropriate setting of m brings the query complexity to $\operatorname{polylog} n$.

Testing $(1,3,2)$-freeness. In this case, we build on the ideas of Newman and Varma [NV24] in which they designed sublinear-query π-freeness testers for general constant-length patterns π for functions $f\colon [n] \to \mathbb{R}$. Their basic tool is to look at the 2-dimensional 'graph' of the function, and approximate it by a 2-dimensional grid of m^2 boxes which are $\{0,1\}$-tagged using what they call Layering and Gridding (for a certain parameter $m \ll n$). Then, using a theorem of Marcus-Tardos [MT04], it can be shown that one may assume that only $O(m)$ of the boxes are nonempty (tagged '1'). Finally, with this reduced number of boxes, they show that the forbidden appearances can be partitioned into a constant number of configurations, each of which is testable by reducing to testing smaller forbidden patterns.

In this work, we follow the same grand scheme of ideas, adapted to $f\colon [n]^2 \to \mathbb{R}$. Now, the 'graph' of the function becomes a 3-dimensional collection of points. We partition the domain into m^2 blocks, and perform Layering and Gridding in a fashion similar to [NV24]. Since there is a corresponding generalization of the Marcus-Tardos theorem [MT04] to Boolean grids of the form $[n]^r$ given by Jang, Nešetřil, and Ossona de Mendez [JNdM21] (in our case, $r := d+1 = 3$), this implies that after $\widetilde{O}(m^2)$ queries, we either find a π-appearance, or we partition the corresponding 3-dimensional graph of the function into a grid $[m]^3$ of boxes with at most m^2 nonempty boxes. Generally, we now analyse the constantly many possible configurations as done in the 1-dimensional

[6] In later works, much more careful and elaborate partitions were used. But, these are also sensitive to the distances between the points forming the appearance.

case, and design sublinear-query testers for each of those configurations. However, while doing so, we face two new difficulties: In the 1-dimensional case, the procedure Gridding results in a grid $[m]^2$ of boxes with $O(m)$ nonempty boxes, and in which every row and every column contains $O(1)$ nonempty boxes (after a simple 'cleaning'). This is crucial to carry out the reduction to testing patterns of smaller size. In our case, the corresponding grid is a grid $[m]^3$ of boxes with $O(m^2)$ nonempty boxes. Hence, on average, every column along any dimension has $O(1)$ nonempty boxes. However, unlike in the 1-dimensional case, we can only guarantee this fact deterministically for every vertical column, but not for horizontal columns. This requires a different strategy altogether.

An additional difficulty was hinted at previously for monotonicity testing. As in [NV24], and the testers for $(1,2,3)$-freeness in this work, most cases have complexity growing with the parameter m, and this is true for the Gridding procedure as well. However, there is one case – namely, when every appearance has all its points in one layer, in which one cannot reduce the testing to monotonicity testing, and one can only guarantee finding an appearance w.h.p. by directly sampling $\mathrm{poly}(n)/\mathrm{poly}(m)$ points. Namely, the query complexity in this case is inversely proportional to m. Optimizing w.r.t. m still only results in an overall query complexity of $\widetilde{O}(n)$. A way to decrease the complexity, which worked in [NV24] for a similar situation, was to use recursion in the above special case, using smaller values of m. However, here, the induced problem on a single layer (on which we would like to perform recursion) is not identical to the original problem, but rather to a test on a partial function on a domain of sublinear size. As explained above, this is not the same as testing a total function as the subdomain is not isomorphic to a rectangular grid anymore. A different kind of recursion has to be worked out, and this results in a complexity of $\widetilde{O}(n^{4/5+o(1)})$.

Lower bounds for testing $(1,3,2)$-freeness. To prove our lower bounds, the high-level idea is to follow Newman, Rabinovich, Rajendraprasad, and Sohler [NRRS19], in which they proved an $\Omega(\sqrt{n})$-query lower bound for *nonadaptively* testing $(1,3,2)$-freeness on the line $[n]$. We define a suitable *intersection search* problem (Theorem 6.3) for two 2-*dimensional* 'monotone arrays' (w.r.t. $\prec$) and show a reduction to to the problem of testing $(1,3,2)$-freeness on the 2-dimensional hypergrid. This reduction is blackbox, and as such, any lower bound for it (nonadaptive or adaptive), implies a corresponding lower bound for testing $(1,3,2)$-freeness. However, unlike in [NRRS19], where the corresponding 1-dimensional problem could be solved adaptively in $O(\log n)$ queries, we show that our 2-dimensional variant requires $\Omega(\sqrt{n})$ queries, even for adaptive randomized algorithms. The latter is shown conceptually by observing that the intersection search problem 'embeds' in it independent instances of a variant of the *birthday problem* (Theorem 6.4).

1.4 Conclusions and open directions

We initiate a systematic study of order pattern freeness of real-valued functions over hypergrids. On the positive side, we design sublinear-query algorithms for testing π-freeness for patterns of length 3 over 2-dimensional hypergrids. We also prove lower bounds to argue that our algorithms are almost optimal. Additionally, we show that the equality between Hamming and deletion distances does not hold for larger patterns in general. This, in turn, implies that current ideas used in the testing literature are insufficient to obtain sublinear-query π-freeness testers for larger patterns.

In the following, we list a few additional interesting open directions that arise from our work.

1. **Testing larger order patterns over high-dimensional hypergrids.** One natural direction is to design π-freeness testers for $\pi \in \mathcal{S}_3$ for hypergrids with dimension larger than 2.

We believe that our techniques might be generalizable to this case and we leave it as an open direction. As noted above, since the Hamming distance is not equal to the deletion distance, in general, for patterns of length 4 over high-dimensional domains, current techniques do not result in sublinear-query testers. We leave open the testing of larger order patterns with respect to the Hamming distance for high-dimensional domains. Note that the question remains open even for the case of 2-dimensional grids.

2. **Testing monotone patterns.** An open question is whether one can obtain efficient testers for monotone patterns of length 4 or more over hypergrids of arbitrary dimension and improve our results. We remark (see Theorem 2.2) that the Hamming distance and the deletion distance are equal for monotone patterns of any length, for functions over hypergrids of arbitrary dimension.

3. **Optimality.** Determining the optimality of our π-freeness testers is a well-motivated problem. It is also important to know whether one can design ER monotonicity testers with better query complexity over high-dimensional hypergrids that work for all fractions of erasures. Additionally, lower bound techniques developed for pattern freeness might find use in proving lower bounds for other related combinatorial problems, as evidenced in the 1-dimensional case [NV21].

4. **Non-permutation patterns.** We consider only order-patterns that are permutations in this paper. It is equally interesting to consider testing freeness of forbidden patterns that are not permutations. In the 1-dimensional case, any order pattern is a permutation, due to the fact that any subdomain of $[n]$ is a total order. However, for $[n]^d$, $d \geq 2$, there are order patterns (of length 3 and higher) that are not permutations. The Hamming distance is not equal to the deletion distance even for such patterns of length 3, in general. We briefly discuss testing freeness of such patterns in Appendix E.2.

1.5 Organization

The paper is organized as follows. Section 2 contains preliminary notations and claims regarding the relationship between the deletion and Hamming distances. Our erasure-resilient and partial function monotonicity testers for real-valued functions over hypergrids are presented in Section 3. Section 4 contains the description of the polylog n-query tester for $(1,2,3)$-freeness. Our $O(n^{4/5+o(1)})$-query tester for $(1,3,2)$-freeness is presented in Section 5. Lastly, Section 6 contains the adaptive and nonadaptive lower bounds for $(1,3,2)$-freeness. Many of the proofs, technical details, and pseudocodes are deferred to the appendices (Appendices A to E). In particular, Appendix D contains the procedures Gridding and Layering that are crucial components of our $(1,3,2)$-freeness testers.

2 Preliminaries

For a function $f\colon [n]^d \to \mathbb{R}$, we denote the range of f by $R(f)$. The elements of the domain $[n]^d$ of f are referred to as *indices* or *points* and the elements of the range $R(f)$ are referred to as *values* or f-values. For an index $x \in [n]^d$, we use x_i to denote the i-th coordinate of x. For two distinct indices $x, y \in [n]^d$, we say that $x \prec y$ if $x_i \leq y_i$ for all $i \in [d]$.

Let $\pi \in \mathcal{S}_k$ be a permutation of length k. The permutation (also called an order pattern or simply a pattern) is monotone if it is either $(1, 2, \ldots, k)$ or $(k, k-1, \ldots, 1)$. It is nonmonotone

otherwise. The function $f\colon [n]^d \to \mathbb{R}$ has a π-appearance if for some points $x^{(1)} \prec \cdots \prec x^{(k)}$, for all $i, j \in [k]$, $\pi(i) < \pi(j)$ implies that $f(x^{(i)}) < f(x^{(j)})$. Namely, the order f induces on $\{f(x^{(i)}) \mid i \in [k]\}$ is identical to the order π induces on $[k]$. For $i \in [k]$, the pair $(x^{(i)}, f(x^{(i)}))$ is called the i-th leg of the π-appearance and the pair $(x^{(\pi^{-1}(i))}, f(x^{(\pi^{-1}(i))}))$ is called the i-leg of the appearance. For example, if x_1, x_2, x_3 are the indices of a $(1,3,2)$-appearance, then $(x_2, f(x_2))$ is the 3-leg and $(x_3, f(x_3))$ is the 2-leg. The function f is π-free if it has no π-appearance. Let P^d_π denote the property of π-freeness of functions $f\colon [n]^d \to \mathbb{R}$. Thus, f is monotone nondecreasing if it satisfies P^d_π for $\pi = (2,1)$.

Most of the algorithms in this paper are for functions over the domain $[n]^2$. We use $x(p)$ and $y(p)$ to denote the x and y coordinates of the point $p \in [n]^2$, respectively.

For ease of presentation, all of our algorithms are designed to have success probability $1 - 1/n^{\Omega(\log n)}$, and we ignore the dependence on polylogarithmic factors in n in their query complexities.

2.1 Deletion and Hamming distances

A major tool in forbidden order pattern testing is the connection between the deletion distance and the Hamming distance.

Definition 2.1 (Deletion and Hamming distances)**.** *Let $f\colon [n]^d \to \mathbb{R}$ and π be an order pattern. The deletion distance of f from being π-free is* $\min\{|S| : S \subseteq [n]^d,\ f|_{[n]^d \setminus S}$ *is π-free$\}$. *Namely, it is the cardinality of the smallest set $S \subseteq [n]^d$ that intersects each π-appearance in f. The Hamming distance of f from being π-free is the minimum of* $\operatorname{dist}(f, f') = |\{i : i \in [n]^d,\ f(i) \neq f(i')\}|$ *over all functions $f' : [n]^d \to \mathbb{R}$ that are π-free.*

For the 1-dimensional case, namely for $f\colon [n] \to \mathbb{R}$ and any $\pi \in \mathcal{S}_k$, the Hamming distance of f to P^1_π is equal to the deletion distance of f to P^1_π. This fact is the basis for every known tester for pattern freeness in the 1-dimensional case. However, for $d \geq 2$ this equality is not true in general. Moreover, for some patterns (and functions), there is an unbounded gap between the two distances. We show that equality between the two distances does hold for some patterns, and, in particular, for all patterns $\pi \in \mathcal{S}_k,\ k \leq 3$.

Claim 2.2. *Let $k \geq 1$ and $\pi \in \mathcal{S}_k$ with $\{\pi(1), \pi(k)\} \cap \{1, k\} \neq \emptyset$. Then the Hamming distance of $f\colon [n]^d \to \mathbb{R}$ to P^d_π is equal to the deletion distance of f to P^d_π.*

The proof of Theorem 2.2 can be found in Appendix E.1.

Corollary 2.3. *Let $\pi \in \mathcal{S}_k,\ k \leq 3$. Let $n, d \geq 1$. The Hamming distance of $f\colon [n]^d \to \mathbb{R}$ to P^d_π is equal to the deletion distance of f to P^d_π.*

Proof. The claim holds for $(1,2) \in \mathcal{S}_2$, $(1,2,3) \in \mathcal{S}_3$, and $(1,3,2) \in \mathcal{S}_3$, by Theorem 2.2. All the other elements of $\mathcal{S}_2$ and $\mathcal{S}_3$ are equivalent to one of these via looking at the domain $[n]^d$ upside-down, or via multiplying all f-values by -1. □

Due to Theorem 2.3, henceforth, for patterns of length at most 3, we simply say that f is ϵ-far from π-freeness without specifying the distance measure.

Unfortunately, a statement analogous to Theorem 2.3 does not hold in general for all patterns of length 4 or higher, even when $d = 2$. This is a significant point of departure from pattern freeness on arrays (i.e., $d = 1$) where such a statement holds for every pattern. We show an example of a

4-pattern π and a function $f\colon [n]^2 \to \mathbb{R}$ for which the multiplicative gap between the Hamming and deletion distances from π-freeness is $\Omega(n)$. This example can be straightforwardly generalized to longer patterns and for larger values of d.

Claim 2.4. *There exists a $\pi \in \mathcal{S}_4$ and a function $f\colon [n]^2 \to \mathbb{R}$ such that the deletion distance of f from π-freeness is 1 and the Hamming distance of f from π-freeness is $\Theta(n)$.*

The proof of Theorem 2.4 can be found in Appendix E.2.

A *matching* of π-appearances in f is a collection of π-appearances that are pairwise disjoint as sets of indices in $[n]^d$. The following claim is folklore and immediate from the fact that the size of a minimum vertex cover of a k-uniform hypergraph is at most k times the cardinality of a maximal matching.

Observation 2.5. Let $\pi \in \mathcal{S}_k$. If $f\colon [n]^d \to \mathbb{R}$ is ϵ-far from being π-free both in Hamming and deletion distances, then there exists a matching of π-appearances of size at least $\epsilon n^d/k$.

We also need the following (folklore) lemma that generalizes the birthday paradox.

Lemma 2.6. *Let N be a set and M be a set of pointwise disjoint k-tuples on N with $k = O(1)$. Then, a uniformly random sample (with repetitions) of $\frac{|N|}{|M|^{1/k}}$ points contains a member of M with probability $\Theta(1)$.*

A direct consequence of Theorem 2.5 and Theorem 2.6 is that for any $\pi \in \mathcal{S}_k$ and any d, if the deletion distance of any $f : [n]^d \to \mathbb{R}$ from P^d_π equals (or is of the order of) the Hamming distance of f from P^d_π, then P^d_π can be tested with sublinear query complexity (in the domain size), as stated below. In particular, this is true for monotone π of any constant size. Our goal, in the rest of the paper, is to improve this much further for $k \leq 3$.

Observation 2.7. Let $\pi \in \mathcal{S}_k$. If for every $f\colon [n]^d \to \mathbb{R}$, the deletion and the Hamming distances of f from P^d_π are equal, then there is a nonadaptive one-sided error ϵ-tester for P^d_π that makes $O(n^{d(1-1/k)}/\epsilon^{1/k})$ queries. In particular, this is true for every monotone k-permutation and every d. Additionally, this is true for every permutation $\pi \in \mathcal{S}_3$ and $d = 2$, implying a $O(n^{4/3}/\epsilon^{1/3})$-query tester.

3 Erasure-resilient monotonicity testing over high dimensions

In this section, we describe our erasure-resilient (ER) monotonicity tester for real-valued functions over hypergrids of arbitrary dimension. Such a tester for 2-dimensional hypergrids is a key subroutine in our pattern freeness testers in Section 4 and Section 5.

Theorem 3.1. *There is a one-sided error nonadaptive δ-ER ϵ-tester for monotonicity of functions $f\colon [n]^2 \to \mathbb{R}$ making $O\left(\log^{O(1)}(n) \cdot \frac{\log^2(1/\epsilon)}{\epsilon(1-\delta)}\right)$ queries.*

Theorem 3.1 asserts the correctness and complexity of an ER monotonicity tester, which serves our purpose for $(1,2,3)$-freeness testing. However, for testing $(1,3,2)$-freeness, we need the ability to test monotonicity of partial functions $f : P \to \mathbb{R}$, where P is an arbitrary subset of $[n]^2$. The subdomain P is known to the tester. An ER monotonicity tester as described above can be used for partial function monotonicity testing. However, P can be arbitrarily small in many cases and the

inverse dependence of the query complexity on $|P|$ (from Theorem 3.1) is not desirable. To address this issue, we design an alternate tester whose query complexity does not depend on $|P|$. We stress that the tester crucially depends on the fact that P is known (unlike in the erasure-resilient case).

Theorem 3.2. *There is a one-sided error nonadaptive ϵ-tester for monotonicity of partial functions $f\colon P \to \mathbb{R}$, where $P \subseteq [n]^2$, with query complexity $O\left(\log^{O(1)}(n) \cdot \frac{\log^2(1/\epsilon)}{\epsilon}\right)$.*

The proofs of Theorem 3.1 and Theorem 3.2 can be found in Appendix A.1 and Appendix A.2.

For using the monotonicity testers described above in the 3-pattern testing, we require more. We need the testers to output not just an arbitrary violation to monotonicity, but rather a "typical" one as we remark below.

Remark 3.3. Let M be a matching of $(1,2)$-pairs on a domain D. Theorem 3.1 describes a tester for monotonicity that finds a violation (p_1, p_2) with p_1 and p_2 being the 1-leg and the 2-leg of some pairs in M (but not necessarily a pair by itself) with constant probability. The tester induces a probability distribution P_M on the 1- and 2-legs of the pair it produces and in particular, a distribution on the 1-leg p_1. It is easy to see that this probability is uniform over the 1-legs of pairs in M.

Remark 3.4. Our monotonicity testers work even when the input function is over a rectangular grid $[n_x]\times[n_y]$, and the dependence of the query complexity on the input size is then polylog$(\max\{n_x, n_y\})$.

Remark 3.5. Our monotonicity testers can be generalized to functions defined on higher-dimensional hypergrids $[n]^d$, for any $d \geq 1$, thereby completing the proof of Theorem 1.5. The details can be found in Appendix A.3.

4 Testing $(1,2,3)$-freeness of functions $f\colon [n]^2 \to \mathbb{R}$

Our main goal in this section is to present a polylog n-query, nonadaptive one-sided error ϵ-tester for $(1,2,3)$-freeness of real-valued functions on the 2-dimensional grid $[n]^2$.

Theorem 4.1. *There exists a nonadaptive one-sided error ϵ-tester for $(1,2,3)$-freeness of functions $f\colon [n]^2 \to \mathbb{R}$ that uses $\log^{O(1)} n$ queries for any constant $\epsilon \in (0,1)$.*

We first present some required preliminaries. Then, we describe a simpler $\widetilde{O}(n)$ tester which is later generalized to the final tester.

4.1 Preliminaries for the $(1,2,3)$-freeness tester

Our goal is to design a one-sided error tester for $(1,2,3)$-freeness. That is, we are only going to reject upon finding a $(1,2,3)$-appearance. Thus, in what follows, we may assume that f is ϵ-far from being $(1,2,3)$-free. By Theorem 2.5, f contains a matching M of $(1,2,3)$-appearances of size $\Omega(\epsilon n^2)$.

4.1.1 Grid of boxes

For a fixed parameter $m \leq n$, let $G_m^{(2)}$ be a partition of $[n]^2$ into $m \times m$ boxes of size n^2/m^2 each, in the natural way. These boxes are arranged naturally in rows and columns in $G_m^{(2)}$. Here, a row refers to a set of points defined by a range of y-coordinates and a column refers to a set of points

defined by a range of x-coordinates. For ease of description, we refer to the direction of increase of the x-coordinate as being from left to right and the direction of increase of the y-coordinate as being from bottom to top of the grid $[n]^2$.

A major tool that we use is to partition M into a relatively small number of certain types of matchings, and try to find a π-appearance in one of the types. This is done as follows. Let $\bar{p} = (p_1, p_2, p_3)$ be a π-appearance in M. We say that $\bar{p}$ is in M_0 if the legs of (p_1, p_2, p_3) all belong to the same box in $G_m^{(2)}$. The point $\bar{p}$ is in M_1 if its legs belong to boxes that are all in the same row (or column) in $G_m^{(2)}$, but not all in the same box. The matching M_2 contains all appearances $\bar{p} = (p_1, p_2, p_3)$ where the boxes containing p_1 and p_2 do not share a row or column in $G_m^{(2)}$ (see Figure 7). Finally, $\bar{p}$ is in M_3 if p_1, p_2 share a row (or a column), and p_2, p_3 share a column (or a row), altogether spanning 2 rows and 2 columns (see Figure 2). With these definitions, the matching M is partitioned into $M = M_0 \cup M_1 \cup M_2 \cup M_3$. We note that the cases where we replace p_1 with p_3 are similar to the cases above. For instance, the case where p_3 is not in the same row or column as p_2 is identical to M_1. Thus, these cases cover all possible configurations of $(1, 2, 3)$-appearances. Since $|M| \geq \epsilon n^2$, it must be the case that

1. either M_0 has cardinality at least $\epsilon n^2 \cdot (1 - \frac{1}{\log n})$
2. or one of M_1, M_2, M_3 has size at least $\epsilon n^2/(3 \log n)$.

We will now show how to develop an algorithm for each case.[7]

4.2 A $\widetilde{O}(n)$ tester for $(1, 2, 3)$-freeness

The tester gets $\epsilon \in (0, 1), n \in \mathbb{N}$ as inputs and has oracle access to a function $f \colon [n]^2 \to \mathbb{R}$. Let $m := \sqrt{n}$, and let $G_m^{(2)}$ be the corresponding 2-dimensional grid of boxes. The tester runs all of these procedures and rejects if any of them finds a $(1, 2, 3)$-appearance.

4.2.1 M_0 is large

We assume that $|M_0| \geq \epsilon n^2 \cdot (1 - \frac{1}{\log n})$. Let $\epsilon^{(i)} = \epsilon \cdot (1 - \frac{1}{\log n})^i$. A uniformly random box, in expectation, contains a matching of $(1, 2, 3)$ appearances of size at least $\epsilon^{(1)} n^2/m^2$. By the reverse Markov inequality, at least $\epsilon/(2 \log n)$ fraction of boxes each contain matchings of cardinality at least $\epsilon^{(2)} n^2/m^2$. That is, if we sample $\Theta(\log^2 n)$ boxes uniformly and independently at random, then with probability at least $1 - n^{-\Omega(1)}$, one of them contains a matching of cardinality at least $\epsilon^{(2)} n^2/m^2$. Therefore, querying all the points in each of these boxes, will be sufficient to find a $(1, 2, 3)$-appearance. The query complexity is clearly $\widetilde{O}(n^2/m^2)$, which is $\widetilde{O}(n)$ by our setting of m.

4.2.2 M_1 is large

We assume that M_1 has cardinality at least $\epsilon n^2/(3 \log n)$. Without loss of generality, we may assume that each of these appearances have all their legs belonging to the same row. There are two main possibilities depending on whether the legs occupy 2 or 3 boxes in the row. Here, we outline the cases where the legs occupy 2 boxes. The case of the legs occupying 3 boxes is similar and is omitted.

[7] It would be more natural to assume that one of the 4 cases has size at least $\epsilon n^2/4$. This will be enough for the $\widetilde{O}(n)$ algorithm, but this is not good enough for the polylog n-query algorithm. Hence, we use this asymmetric assumption.

Further, it could be that the 1- and 2-legs belong to the same box or the 2- and 3-legs belong to the same box. We only consider the former case as the arguments to deal with the latter case are analogous.

For two domain points $u, v \in [n]^2$, the y-spacing between u and v is the distance between the y-coordinates of u and v. The matching of $(1,2,3)$-appearances in which the 1- and 2-legs are in the same box, but the 3-leg is in a different box, can be further split into these appearances for which the y-spacing between 2- and 3-legs is more than that between the 1- and 2-legs. We denote this submatching by M_{11}, and denote by M_{12} the submatching for which the y-spacing between 1- and 2-legs in the $(1,2,3)$-appearance is more than that between the 2- and 3-legs (see Figure 1).

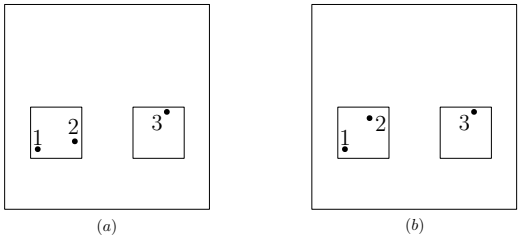


Figure 1: (a) M_{11}: $\Delta_y(\text{2-leg}, \text{3-leg}) \geq \Delta_y(\text{1-leg}, \text{2-leg})$ (b) M_{12}: $\Delta_y(\text{2-leg}, \text{3-leg}) < \Delta_y(\text{1-leg}, \text{2-leg})$

The easy case: M_{11} is large. The subcase where M_{11} is large (i.e., $|M_{11}| \geq \epsilon n^2/(c \log n)$ for some absolute constant c) is easy to handle. At a high level, we can partition M_{11} into logarithmically many buckets according to the y-spacing between the 2- and 3-legs, much like in [NRRS19]. For the right bucket, it is easy to first sample a box B containing *sufficiently many* 3-legs and then run a $(1,2)$-freeness tester to find a corresponding $(1,2)$-appearance in the same row as B. The existence of many such $(1,2)$-appearance is guaranteed by the 'median' argument. A detailed discussion is deferred to the appendix (Appendix B.1.1).

The hard case: M_{12} is large. Suppose that the cardinality of M_{12} is at least $\epsilon' n^2$, where ϵ' is $\epsilon/(c \log n)$ for some absolute constant c. The reason that this case is harder is because sampling an intended 3-leg in B as above would determine a relatively small region for the 2-leg, but the 1-leg could be in a much larger region, and so, the corresponding $(1,2)$-appearances have too small a density. For this case, we will need to resort to the additional 'typicality' property of the output of the $(1,2)$-freeness tester.

We start by sampling one box B. With probability at least $\epsilon'/2$, a uniformly random box has at least $(\epsilon'/2) \cdot (n^2/m^2)$ many $(1,2)$-appearances. Thus, with probability at least $1 - n^{-\Omega(1)}$, one out of $\widetilde{O}(1/\epsilon)$ sampled boxes satisfies this. We condition on sampling such a box B.

Let R be the row containing B and let R' be the part of the row R to the right of B. For each $k \in [\log n]$, we consider equipartitioning R into $n/(2^k m)$ horizontal strips, each being a grid isomorphic to $[n] \times [2^k]$. Let $k' \in [\log n]$ be the scale that contains the largest number of the $(1,2,3)$-appearances whose 2- and 3-legs are on adjacent strips. That is, the row R restricted to this scale contains a matching M' of $(1,2,3)$-appearances of size at least $(\epsilon'/(2 \log n)) \cdot (n^2/m^2)$, where the 2- and 3-legs are on adjacent strips (but the corresponding 1-leg could be anywhere). Hence, a random strip of this scale will contain at least $\epsilon'/(2 \log n)$ fraction of points that are the 2-legs of appearances of that scale in M_{12}.

Now the argument is as follows: we first apply the monotonicity tester with proximity parameter[8] $\Theta(\epsilon'/\log n)$ to B to find a $(1,2)$-appearance corresponding to $(1,2,3)$-appearances from M_{12} of the fixed scale above. Further, by the typicality feature stated in Theorem 3.3, this 2-leg is picked uniformly at random from all 2-legs in the submatching of M_{12} restricted to the above fixed scale, and to the strip where the 2-leg is. In particular, the strip that contains the point picked as the 2-leg is a random strip among the strips of the above fixed scale, and the value of the 2-leg is below the median of all 2-legs of that scale in that strip.

Assuming the above, now the situation is much simpler: since by the assumption above the corresponding strip contains at least $\epsilon'/(4\log n)$ fraction of points that are 2-legs of corresponding $(1,2,3)$-appearances, and assuming the median feature for the actual picked 2-leg, above, there are at least $\epsilon'/(8m\log n)$ fraction of points in the next strip of the same fixed scale and to the right of B that are the corresponding 3-legs, and with value above the value of our picked 2-leg. Selecting a random point in this strip will find one resulting in a $(1,2,3)$-appearance. Again, working out the details (and using amplification to reach a high success probability) results in a $\widetilde{O}(m/\epsilon^2)$-query algorithm for this case. The pseudocode for this case can be found in the appendix (Algorithm 2).

4.2.3 M_2 is large

M_2 is the matching of $(1,2,3)$-appearances where the boxes containing the 1- and 2-legs do not share a row or column. The discussion of the case where M_2 is large, i.e., $|M_2| \geq \epsilon n^2/(3\log n)$, is deferred to the appendix (Appendix B.2).

4.2.4 M_3 is large

In this case, the only possibilities are that the three boxes containing the legs of a $(1,2,3)$-appearance form a Γ-shape or an inverted Γ-shape, as shown in Figure 2. The treatment of both cases is similar, so we assume that a majority of the $(1,2,3)$-appearances in M_3 form a Γ-shape: that is, the $1,2$-legs are on the same column, and the $2,3$-legs are on the same row, as depicted in Figure 2(a).

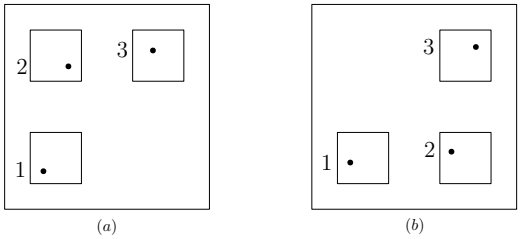


Figure 2: M_3: (a) $(1,2,3)$-appearance in a Γ-shape (b) $(1,2,3)$-appearance forming an inverted Γ-shape

Consider a partition of M_3 as follows. For $\ell = 0,1,\ldots,\log n$, let $(p_1,p_2,p_3) \in M_3$ belong to $M^{(\ell)}$ for the smallest ℓ such that $x(p_2) - x(p_1) < 2^\ell$, where $x(p)$ denotes the x-coordinate of the point p.

Recall that by our assumption, $|M_3| \geq \epsilon n^2/(3\log n)$. Let $\ell^* \in \{0,1,\ldots,\log n\}$ be such that $|M^{(\ell^*)}| = \Omega(\epsilon n^2/(\log^2 n))$. We first consider the case when $\ell^* = 0$. That is, for $\Omega(\epsilon n^2/(\log^2 n))$

[8] A monotonicity tester T with *proximity parameter* ϵ distinguishes the case that the queried function f is monotone from the case that f is ϵ-far from monotone, with probability at least $2/3$.

many $(1,2,3)$-appearances in $M^{(\ell^*)}$ the x-coordinate of the 1- and 2-legs are equal. Now, a uniformly random box B will contain the 2-legs of $\Omega(\epsilon n^2/(m^2 \log^2 n))$ many $(1,2,3)$-appearances from $M^{(\ell^*)}$, in expectation. In other words, with probability $\widetilde{\Omega}(\epsilon)$, a uniformly random box will contain the 2-legs of $\Omega(\epsilon n^2/(m^2 \log^2 n))$ many $(1,2,3)$-appearances from $M^{(\ell^*)}$. Thus, in $\widetilde{\Theta}(1/\epsilon)$ iterations, with probability $1 - n^{-\Omega(1)}$, there exists a sampled box B that contains the 2-legs of $\Omega(\epsilon n^2/(m^2 \log^2 n))$ many $(1,2,3)$-appearances from $M^{(\ell^*)}$. In the following, we condition on such a box B being sampled.

For this box B, let R', as before, denote the row containing B when restricted to the boxes to the right of B. We can see that R' is $\Omega(\epsilon/(m \log^2 n))$-far from $(1,2)$-freeness, by virtue of the 2- and 3-legs of the $(1,2,3)$-appearances. Thus, running the $(1,2)$-freeness tester with proximity parameter $\Omega(\epsilon/(m \log^2 n))$ for $\widetilde{\Theta}(m/\epsilon)$ iterations, finds $\widetilde{\Theta}(m/\epsilon)$ many such $(2,3)$-appearances, with probability $1 - n^{-\Omega(\log n)}$.

For each of the $\widetilde{\Theta}(m/\epsilon)$ many $(2,3)$-appearances found above, the 2-leg is a uniformly random point among the 2-legs of B, by Theorem 3.3. Now, let C be the column containing box B and consider a partition of C into subcolumns where each subcolumn, in this case, consists of points with the same x-coordinate. These n/m columns cover the set of all 2-legs in B. Let (p_2, p_3) be one such $(2,3)$-appearance returned above. Since p_2 is a uniformly random point among the 2-legs of B, there are, in expectation, $\Omega(\epsilon n/(m \log^2 n))$ many 2-legs in the subcolumn of B containing p_2. From this, one can conclude that, with probability $\Omega(\epsilon/(m \log^2 n))$, (1) the point p_2 belongs to a subcolumn that has $\Omega(\epsilon n/(m \log^2 n))$ many 2-legs and (2) that $f(p_2)$ is at least the median value of 2-legs in its subcolumn. Since we have $\widetilde{\Theta}(m/\epsilon)$ many such $(2,3)$-appearances, with probability at least $1 - n^{-\Omega(\log n)}$, one of them must satisfy the above conditions. Let (p_2', p_3') denote that $(2,3)$-appearance. Now, it must be the case that there are $\Omega(\epsilon n/(m \log^2 n))$ many 1-legs in the subcolumn of p_2' that can, together with p_2', result in a $(1,2,3)$-appearance. Since we sample $\widetilde{\Theta}(m/\epsilon)$ many points from the subcolumn of each such 2-leg sampled, we succeed in finding a $(1,2,3)$-appearance with probability at least $1 - n^{-\Omega(\log n)}$.

The argument for $\ell^* \geq 1$ is very similar to the one and is omitted for brevity. The query complexity of a procedure resulting from the above idea is $\widetilde{O}(m^2 \operatorname{poly}(1/\epsilon))$, and its pseudocode can be found in the appendix (Algorithm 4).

Remark 4.2. In the description above, only the step that uses sampling to detect the 1-leg is adaptive, as it is performed w.r.t. the $(2,3)$-appearances that are found. However, since such an appearance is guaranteed to be found with high probability, this adaptivity is not algorithmically needed. Hence the whole algorithm can be made nonadaptive with essentially the same complexity.

4.3 A recursive algorithm

Our goal here is to improve the $\widetilde{O}(n)$ tester to a polylog n query algorithm and prove Theorem 1.2.

Let $\epsilon^{(i)} = \epsilon \cdot (1 - \frac{1}{\log n})^i$. Let $m = \log^2 n$. The idea is to use recursion in the case that $\epsilon^{(1)}$ fraction of $(1,2,3)$-appearances each have all their legs in one box in $G_m^{(2)}$. This corresponds to the case that M_0 has large size in the description in the preceding sections. Consider a collection of $t = t^{(1)} = \Theta(\frac{\log^3 n}{\epsilon^{(1)}})$ boxes sampled uniformly and independently at random from $G_m^{(2)}$. Let X denote the size of the matching of $(1,2,3)$-appearances restricted to these boxes. We know that $\mathbf{E}[X] = t \cdot \frac{\epsilon^{(1)} n^2}{m^2}$. By the Hoeffding's inequality, we know that $X \geq t \cdot \frac{\epsilon^{(2)} n^2}{m^2}$, with probability at least $1 - n^{-\Omega(1)}$. Let $B_1, \ldots, B_t$ denote the boxes sampled above and let S denote the union of these boxes in $G_m^{(2)}$. From the above, we can conclude that the region S is $\epsilon^{(2)}$-far from being

$(1,2,3)$-free. We recursively call our algorithm on S. The important point is that we only need to consider $(1,2,3)$-appearances that are fully contained within the boxes $B_1,\dots,B_t$ and we can ignore any appearance that has its legs belonging to different boxes. In other words, we are, in a sense, solving t disjoint subproblems in parallel.

First, we consider a gridding of each of these boxes into $m \times m$ subboxes. Suppose that at least $\epsilon^{(2)}/(3\log n)$ fraction of all the $(1,2,3)$-appearances in S are such that their legs all belong to subboxes in the same row but not all in the same subbox (as in the case when M_1 is large). It must be the case that there exists one box B_i such that at least $\epsilon^{(2)}/(3t\log n)$ fraction of all the $(1,2,3)$-appearances have their legs belonging to rows of subboxes of B_i. Thus, running the procedure for the case that M_1 is large on each box $B_i, i \in [t]$ separately with parameter $\epsilon^{(2)}/(3t\log n)$ will find such a $(1,2,3)$-appearance with an overall success probability of $1-n^{-\Omega(1)}$. The number of queries is $\widetilde{O}(m^2 \cdot \mathrm{poly}(1/\epsilon^{(2)}))$. The other non-recursive cases can be dealt with in this same fashion.

For the recursive case, we sample $t^{(2)} = \Theta(\frac{\log^3 n}{\epsilon^{(2)}})$ subboxes uniformly and independently at random from the region S and as above, a Hoeffding's bound will guarantee that the union of these subboxes is $\epsilon^{(3)}$-far from being $(1,2,3)$-free.

The base level of the recursion is when the region has size $\Theta(\log^4 n \cdot \mathrm{poly}(1/\epsilon))$, in which case, we query the entire region. Note that at the i-th recursion level, for $i \geq 1$, the size of the domain becomes $(n^2/m^{2i}) \cdot t^{(i)} = (n^2/m^{2i}) \cdot \Theta(\frac{\log^3 n}{\epsilon^{(i)}})$. Setting $m = \log^2 n$ bounds the recursion depth to at most $\log n$. The proximity parameter at the base level is ϵ/c for some absolute constant c. The total query complexity in the non-recursive cases at the i-th recursion level is $\widetilde{O}(m^2 \cdot \mathrm{poly}(1/\epsilon^{(i)}))$ for all $i \geq 1$. Hence, the overall query complexity of the recursive algorithm is $\mathrm{polylog}\, n \cdot \mathrm{poly}(1/\epsilon)$.

Remark 4.3. As in Theorem 4.2, this algorithm can also be made nonadaptive.

This completes the proof of Theorem 1.2, which we restate here for convenience.

Theorem 4.4. *There exists a one-sided error nonadaptive ϵ-tester with query complexity $\log^{O(1)} n$ for $(1,2,3)$-freeness of functions $f\colon [n]^2 \to \mathbb{R}$ for constant $\epsilon \in (0,1)$.* □

5 Testing $(1,3,2)$-freeness

In this section we develop a sublinear-time tester for $(1,3,2)$-freeness and prove Theorem 1.4, where $\pi = (1,3,2)$ is the only forbidden order in $\mathcal{S}_3$, other than $(1,2,3)$, up to order-isomorphism. This case turns out to be harder than testing $(1,2,3)$-freeness, much like the difference in the 1-dimensional case between the respective patterns. In particular, the sublinear-query tester we develop is *adaptive*, as expected in view of the lower bound on nonadaptive testers even for the 1-dimensional case. Here, we need a different set of tools, following [NV24].

5.1 Preliminaries

Let $f\colon [n]^2 \to \mathbb{R}$. The set of values of f along with the 2-dimensional domain naturally defines a 3-dimensional box. This grid, the "graph-grid" is isomorphic to $[n]^2 \times R(f)$ containing n^2 f-points of the form $(p, f(p))$, $p \in [n]^2$. For a parameter $m \leq n$, let $G_m^{(2)}$ be the 2-dimensional $m \times m$ grid of boxes as defined in previous sections. One can then define a coarse 3-dimensional $m \times m \times m$ grid $G_m^{(3)}$ that partitions $[n]^2 \times R(f)$. In particular, $G_m^{(3)}$ partitions the set of f-points, $\{(p, f(p)) : p \in [n]^2\}$ into m^3 parts. Let $I \subseteq [n]$ be an interval belonging to the partition of x-coordinates in $G_m^{(3)}$ above. The region of the grid-graph with x-coordinates in the range I is then referred to as an *x-slice*. Each

x-slice contains m^2 boxes. One can define y- and z-slices similarly. z-slices are also referred to as *layers*. The intersection of an x-slice and a y-slice is an *xy-column* and contains m boxes. One can define yz and xz-columns similarly. We say that a box $B \in G_m^{(3)}$ is nonempty if it contains a point $(p, f(p))$ and empty otherwise.

We use [JNdM21, Theorem 2.2], restated for our case, that generalizes the Marcus-Tardos Theorem [MT04] as follows.

Theorem 5.1. *[JNdM21] For any integers k, m, t such that $k \leq m$, there is a $\tau_t(k)$ such that if a function $g\colon [m]^t \to \{0,1\}$ evaluates to 1 on more than $\tau_t(k) \cdot m^{t-1}$ entries, then, for any k-sized subposet π_k of the t-dimensional grid $[k]^t$, there is a collection of k 1-entries in $[m]^t$ that form an isomorphic copy of π_k.*

We will use Theorem 5.1 with $k = 3$ (corresponding to the forbidden 3-pattern) and $t = 3$ corresponding to the 3-dimensional Boolean grid of boxes that corresponds to the graph of the function $f\colon [n]^2 \to \mathbb{R}$, in a similar way that $t = 2$ was used in [NV24] for functions on the line. Namely, we view $G_m^{(3)}$ with its f-points as a *Boolean* grid isomorphic to $[m]^3$ where a point (box) is 0 if it is empty and 1 otherwise.

The basic tool that allows this test is that the Hamming distance and the deletion distance are equal for the property of $(1,3,2)$-freeness by Theorem 2.3. Hence, in what follows, we may assume that f is ϵ-far from $(1,3,2)$-free, and contains a matching M of $(1,3,2)$-appearances with $|M| = \Omega(\epsilon n^2)$, since we are going to develop a one-sided error tester for this property. We note that any $(1,3,2)$-appearances in f corresponds to a $(1,3,2)$-appearance in the graph-grid (but not necessarily the other way, as in previous sections).

5.2 A $\widetilde{O}(n)$ tester for $(1,3,2)$-freeness

Recall that a trivial deterministic tester for $(1,3,2)$-freeness takes $O(n^2)$ queries. A baseline non-adaptive sublinear-query tester following Theorem 2.6 makes $O(n^{4/3})$ queries. Our aim in this section is to improve this by designing an $\widetilde{O}(n)$-query tester.

Theorem 5.2. *There exists a one-sided error ϵ-tester with query complexity $\widetilde{O}(n)$ for $(1,3,2)$-freeness of functions $f\colon [n]^2 \to \mathbb{R}$ for constant $\epsilon \in (0,1)$.*

5.2.1 High-level outline

Let $m := \sqrt{n}$. We consider the 2-dimensional grid of boxes $G_m^{(2)}$ that partitions the domain $[n]^2$. As in Section 4, we can partition the matching M into a constant number of submatchings according to the configurations that the $(1,3,2)$-appearances form. Moreover, the cases in which all the legs share a row, or share a column, are treated identically as in Section 4, and will not be discussed here.

The same is true for the cases of the partial matchings in which the 1-leg in every appearance does not share a column or row with the 2- or 3-legs (analogous to the cases where M_2 is large in Section 4).

The harder cases are when the 1-leg and the 3-leg share a column or a row (see Figure 3), while the 2-leg could be in a shared row (column) or in a third row (column). The harder cases among these is when the 2-legs share a row or a column with the 3-leg. There are two such configurations – the Γ-shape (Figure 3(a)), and an inverted Γ-shape (Figure 3(b)). The treatment of these two

cases is similar, so we only describe the case in which there is a large matching M_Γ of size $\Omega(\epsilon n^2)$ of Γ-shape $(1,3,2)$-appearances (i.e., the case shown in Figure 3(a)). The cases in which the 2-leg does not share a column or row with the 3-leg (see Figure 3(c) and Figure 3(d)) are similar.

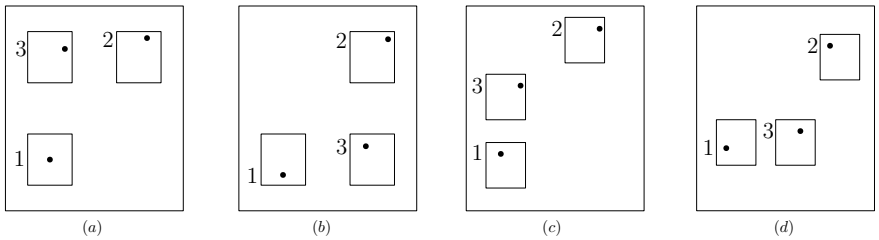


Figure 3: (a), (b) The harder cases – Boxes in $G_m^{(2)}$ containing the legs of a $(1,3,2)$-appearance forming a (a) Γ-shape, and (b) an inverted Γ-shape. (c), (d) The easier cases – Box in $G_m^{(2)}$ containing the 2-leg is in a row and column different from that of the box containing the 3-leg.

The high level description: Let $m := \sqrt{n}$.

1. **Gridding:** We first perform $\mathsf{Gridding}(S, I, m, \beta)$ (detailed in Appendix D.2) with $S := [n]^2$, $I := (-\infty, \infty)$, $m := \sqrt{n}$, and $\beta = \Theta(1)$. The algorithm $\mathsf{Gridding}(S, I, m, \beta)$ uses $\mathsf{Layering}(S, I, m)$ (detailed in Appendix D.1) as a subroutine, which is used to obtain a "nice" partition of $R(f)$ into $m' \leq 2m = \Theta(\sqrt{n})$ layers (see Theorem D.2). $\mathsf{Layering}$ is carried out by making $O(m' \cdot \log^4 n) = \widetilde{O}(m)$ queries. Via $\mathsf{Gridding}(S, I, m, \beta)$, we obtain the 3-dimensional grid $G_m^{(3)}$ that partitions the graph-grid $[n]^2 \times R(f)$ into a set $\mathcal{B}_3$ of $\Theta(m^3)$ boxes with "well-controlled" densities (see Theorem D.4). Each subbox $B \in \mathcal{B}_3$ is tagged by $\mathsf{Gridding}(S, I, m, \beta)$ as *empty* if it contains no sampled points, and *nonempty* otherwise. Further, a nonempty box B is tagged as *dense* by $\mathsf{Gridding}$ if it estimates that B contains $\Theta(n^2/m^2) = \Theta(n)$ f-points. Note that the projection of any B on the domain $[n]^2$ is of size $\Theta(n^2/m^2)$. Hence, *dense* means that a constant fraction of the points in the projection of B on the domain are indeed inside B. Otherwise, B is *not dense*. We prove that our estimates are accurate enough with high probability, and hence assume that the *dense* boxes are indeed dense.

   We note that although there are $\Theta(m^3) = \Theta(n^{3/2})$ boxes in $G_m^{(3)}$, the process will only use $O\left(m \cdot \log^4 n + \frac{\log^4 n}{\beta^2} \cdot 4m^2\right) = \widetilde{O}(m^2) = \widetilde{O}(n)$ queries, and this process works as intended with high probability (at least $1 - 1/n^{\Omega(\log n)}$).

2. As a result of $\mathsf{Gridding}$, we either find $\omega(m^2)$ nonempty boxes, or find out $O(m^2)$ dense boxes that cover all but a negligible fraction of the f-points. In the first case, Theorem 5.1 will guarantee the existence of a $(1,3,2)$-appearance in the already sampled points, in which case we stop and reject. Otherwise, we move on to the next item.

3. If we reach here, we have $O(m^2) = O(n)$ dense boxes in $\mathcal{B}_3$. We note that every slice (x-, y-, or z-slice) contains $O(m)$ dense boxes, since, by averaging, we ignore other cases. By the same reasoning, every xy-column contains $\Theta(1)$ dense boxes (but this is not necessarily true for xz-columns or yz-columns).

We ignore all f-points outside these boxes and will show how to find a $(1,3,2)$-appearance with legs in the dense boxes. Since we ignore an insignificant fraction of the f-points, there is still a matching M' of size $\Theta(\epsilon n^2)$ of Γ-shaped $(1,3,2)$-appearances among the remaining points. We now partition M' into a constant number of submatchings, according to the configuration the appearances form in $G_m^{(3)}$. Each case will be treated separately, as explained in the next section, in $\widetilde{O}(n)$ queries.

5.2.2 Finding a $(1,3,2)$-appearances in a large Γ-shaped matching

We assume in what follows that the matching M of Γ-shaped $(1,3,2)$-appearances is of size $\Omega(\epsilon n^2)$. A similar treatment can be carried out when the matching of inverted-Γ-shaped appearances is large (by possibly switching the x-coordinates and the y-coordinates in the discussion), and for shapes as in Figure 3(c) and Figure 3(d).

1. **M contains $\Omega(\epsilon n^2)$ appearances, where the legs of each appearance share a layer (that is, a z-slice)**

   In this case, a uniformly random layer will have $\Omega(\epsilon n^2/m)$ appearances in expectation. We choose a random layer S_i, $i \in [m]$. Then, with probability $\Omega(\epsilon)$, it will contain $\Omega(\epsilon n^2/m)$ appearances. Thus, sampling $\widetilde{\Theta}(\frac{n^{4/3}}{m^{4/3}\epsilon^{1/3}}) = \widetilde{O}(n/\epsilon^{1/3})$ points uniformly and independently at random, will find an appearance with probability $1 - n^{-\Omega(\log n)}$, by Theorem 2.6.

2. **M contains $\Omega(\epsilon n^2)$ appearances, each of which has its legs on two layers**

   - Most appearances have their 1-leg in a layer, and the $2,3$-legs in a different layer. See Figure 4.

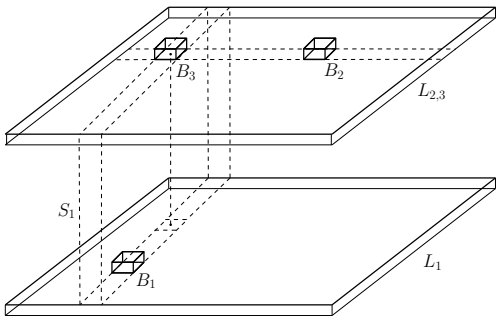


Figure 4: A typical occurrence: 1-leg on layer L_1 and $2,3$-legs are on layer $L_{2,3}$

In this case, a typical dense box $B \in G_m^{(3)}$ will contain $N = \Omega(\epsilon n^2/m^2)$ 3-legs of $(1,3,2)$-appearances in M. Let $M(B_3)$ be this set of N appearances. We choose such a box B_3, and assume it is "typical" (an event which will occur with probability $\Omega(\epsilon)$). Choosing B_3 specifies the layer $L_{2,3}$ in which the corresponding 2-legs of the appearances in $M(B_3)$ lie, and the x-slice (S_1 in the figure) that should contain the 1-leg.

We note that the corresponding x-slice S_1 should contain N many 1-legs that correspond to the 3-legs in B_3. However, we do not know the specific B_1 and B_2 that contain the $1,2$-legs.

We choose a uniformly random point p_3 in B_3. With probability $\Omega(\epsilon)$, it will be a corresponding 3-leg, and with probability $1/4$, given that it is a 3-leg, will have x-coordinate $x(p_3)$ that is above but close to the median of the x-coordinates of 1-legs in S_1 that correspond to the N appearances in B_3. Assuming that this happens, there are $\Omega(\epsilon n^2/m^2)$-appearances whose 3-leg in B_3 has x-coordinate at least $x(p_3)$. Each such 3-leg, together with the corresponding 2-leg, forms a $(3,2)$-appearance in $L_{2,3}$, and to find one, we use our monotonicity tester with distance parameter $\Omega(\epsilon/m)$, restricted to the points with x-coordinate larger than $x(p_3)$. Once such an appearance (p_3^*, p_2) is found, the slice S_1 should contain $N/2$ points p_1 that have x-coordinates lower than $x(p_3)$ and are 1-legs of appearances in M. Namely, a random point satisfies this with probability $\Omega(\epsilon/m)$. Then, sampling $\tilde{\Theta}(m/\epsilon)$ such points will find a point p_1 that together with p_3^*, p_2 forms a $(1,3,2)$-appearance.

The resulting query complexity for this case is $\widetilde{O}(m/\epsilon)$, which is the query complexity of running the monotonicity tester (see Theorem 3.2) with parameter $\Omega(\epsilon/m)$.

- Most appearances have their $1,2$-legs in a layer, and the 3-legs in a different layer (see Figure 5). We note that this is the only other possibility for a Γ-shape to have its two legs over two layers.

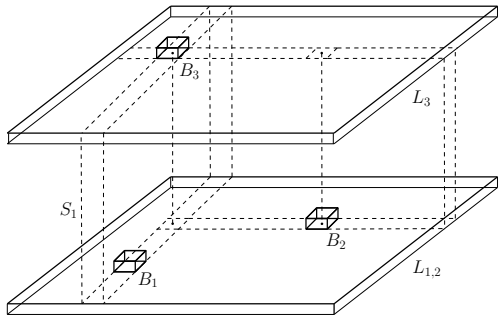


Figure 5: A typical occurrence: $1,2$-legs on layer $L_{1,2}$ and 3-legs are on layer L_3

In this case, we present two solutions. The first, described in **A1** below, is conceptually easier and good enough for the $\widetilde{O}(n)$-algorithm. Its drawback is that its query complexity depends inversely on m and hence requires a relatively large value of m. The second is proportional to m and is therefore better for small values of m which will allow for better recursion.

In both cases we choose a random layer $L_{1,2}$, which, with probability $\Omega(\epsilon)$, will contain $N = \Omega(\epsilon n^2/m)$ $1,2$-legs of the appropriate N appearances $M(L_{1,2})$ in M.

A1: Next, we choose random B_1, B_2 that should contain the corresponding $1,2$-legs of such appearances, respectively. As there are $O(m^2)$ pairs of boxes, in expectation, a uniformly random pair of boxes will contain $\Omega(\epsilon n^2/m^3)$ such appearances whose corresponding pairs of $1,2$ legs are in B_1, B_2 respectively. Now, fixing B_1, B_2, the xy-column containing B_3 is fixed, and by our assumption, contains $\Theta(1)$ dense boxes. Now, there are two ways to find a $(1,3,2)$-appearance in the corresponding boxes. We focus on the constant number of possible B_3, which, together with the fixed B_1, B_2 have $\Theta(n^2/m^2)$ points in total and contains $\Omega(\epsilon n^2/m^3)$ many π-appearances. Hence, by Theorem 2.6, it is enough to sample $O(\frac{n^2}{m^2} \cdot (\frac{m^3}{\epsilon n^2})^{1/3}) = O(\frac{n^{4/3}}{\epsilon^{1/3} m})$ points uniformly and independently at

random from this union of boxes in order to obtain a π-appearance with high probability. The sample complexity is $O(n^{5/6})$ for constant ϵ by our setting of m.

A2: We start as described above, except that the order of the sampling is different. After choosing $L_{1,2}$ we first choose B_1 at random in $L_{1,2}$, which will contain, in expectation, $\Omega(\epsilon n^2/m^2)$ many 1-legs from the matching M. Now, before choosing B_2, we first sample $\Theta(1/\epsilon)$ points from B_1. We expect that at least one of these points is from the set of 1-legs above, and further, that this point has value close to the median value of the 1-legs.

Assuming we have chosen such a p_1 successfully, let M_H contain all appearances whose $1,2$-legs are in $L_{1,2}$ and the 1-leg value is above $f(p_1)$. Since we assume that $f(p_1)$ is close to the median, we conclude that $|M_H| = \Omega(\epsilon n^2/m^2)$. We now choose B_2, and correspondingly B_3 (the latter from constantly many possibilities in its xy-column). Since B_2 is chosen at random, we expect that there is a matching $M_H^1 \subseteq M_H$ of size $\Omega(\epsilon n^2/m^3)$ whose $3,2$-legs are in B_3, B_2, respectively.

We now apply our partial function monotonicity tester (Theorem 3.2) in B_2, B_3 but restricted to points of f-values above $f(p_1)$, with distance parameter ϵ/m, to find such a $(3,2)$-appearance (p_3, p_2). This can be done in $\widetilde{O}(m/\epsilon)$ queries, and moreover, according to Theorem 3.3, the point p_3 is essentially a random point in M_H^1. In particular, its x-coordinate is larger than the x-coordinates of at least half the points in M_H^1. In turn, this implies that a random point in B_1 is likely to be a 1-leg of an appearance in M_H^1, but with x-coordinate smaller than $x(p_3)$ with probability $\Omega(1/m)$. Hence sampling $\widetilde{\Theta}(m)$ points will find such a point p_1^* with high probability. Thus, (p_1^*, p_3, p_2) form a $(1,3,2)$-appearance.

We end this case by noting that the discussion above assumed that we knew such a "successful" p_1 to begin with. Since we do not know the correct p_1 out of our sample, we try for all possible sampled points. Hence, the total query complexity in this case is $\widetilde{O}(m/\epsilon)$.

3. M **contains $\Omega(\epsilon n^2)$ appearances, each with its legs on three layers.** This case, for a Γ-shaped appearance, is depicted in Figure 6.

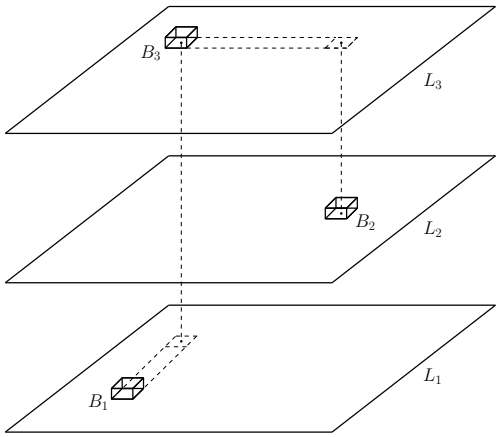


Figure 6: A typical occurrence: $1,2,3$ legs on layer L_1, L_2, L_3 respectively

In this case, a typical x-slice S_1 will contain $N = \Omega(\epsilon n^2/m)$ many 1- and 3-leg pairs from

appearances in M. With constant probability, a random dense box B_1 in S_1 and a random index $p_1 \in B_1$ will have $x(p_1)$ below the median of the x-coordinates of the 1-legs of these N appearances. Assuming this happens, any $(3,2)$-appearance above the layer L_1 and with x-value larger than $x(p_1)$ will result in a $(1,3,2)$-appearance with p_1. Thus, using the monotonicity tester, this can be found in $\widetilde{O}(m/\epsilon)$ queries as the corresponding distance parameter is ϵ/m. This is also the total query complexity for this case.

Since all of these cases, as well as the Layering and Gridding are done using $\widetilde{O}(n)$ queries, the claim on the query complexity of the test follows. This proves Theorem 5.2. We remark that the queries made in Cases 2 and 3 above are adaptive, as they depend on which boxes are marked as dense by the Gridding phase.

5.3 Improving the query complexity to $\widetilde{O}\left(n^{\frac{4}{5}+o(1)}\right)$

In this section, we discuss the ideas to obtain our improved $(1,3,2)$-freeness tester and prove Theorem 1.4.

Proof of Theorem 1.4. From our discussion in the preceding section, the query complexity of all cases except Case 1 (where all legs are in a single layer), is upper bounded by $\widetilde{O}(m^2\,\mathrm{poly}(1/\epsilon))$. The query complexity of Case 1 has an inverse dependence on m, and this is the case where recursion will help improve the complexity.

We focus on Case 1. Namely, we assume Case 1 happens at the top level of the recursion, and that we have sampled a layer L containing $\Omega(\epsilon n^2/m)$ appearances. Using the fact that the number of dense boxes in every layer is $O(m)$, we restrict our attention to the function restricted to the part of the domain defined by the $O(m)$-many xy-columns defined by these dense boxes in the current layer we focus on. The challenge in the recursive call is in performing Layering and Gridding, since we do not have a rectangular domain anymore. We instead do the following. We first perform Layering on this restricted function and form $O(m)$ layers. Now, we perform Gridding on the domains of each of the $O(m)$-many $\frac{n}{m}\times\frac{n}{m}$ sized rectangular subgrids separately with the same gridding parameter m. This results in $O(m^3)$ dense boxes. The overall query complexity for these steps is $\widetilde{O}(m^3\,\mathrm{poly}(1/\epsilon))$. In addition, within the same query complexity, all cases except Case 1, can be done with respect to the function restricted to the layer L.

Recursing to a depth of d in the above fashion, the query complexity for Gridding and all cases except Case 1 at the lowest level of recursion is $\widetilde{O}(m^{d+2}\,\mathrm{poly}(1/\epsilon_d))$, where ϵ_d is the distance parameter at that level. Since the distance parameter drops by a constant factor in each successive level of recursion, we have $\epsilon_d = \epsilon/c^d$ for some constant c.

At the d^{th} recursion level, if we are in Case 1, we restrict our attention to $O(m^{d+1})$ dense boxes all in the same layer, where each dense box has asymptotically $\Theta(n^2/m^{2d+2})$ many points. The $(1,3,2)$-appearances that we look for are present in an ϵ_d-fraction of these points. Thus, by Theorem 2.6, making $\widetilde{O}\left(\frac{n^{4/3}}{m^{\frac{2d+2}{3}}}\cdot\frac{1}{\epsilon_d^{1/3}}\right)$ queries will enable us to find these appearances with high probability. Setting m to be $n^{4/(5d+8)}$, and plugging in the expression above results in $\widetilde{O}\left(m^{\frac{4d+8}{5d+8}}\cdot\mathrm{poly}(1/\epsilon_d)\right)$ complexity. Finally, the total number of nodes in the recursion tree is at most $(\widetilde{\Theta}(1/\epsilon_d))^d$ and the number of queries in each node is upper bounded by the work done at the leaf nodes. Therefore, the overall query complexity for a d-level recursive procedure is $O(m^{\frac{4d+8}{5d+8}}\cdot((\mathrm{polylog}\,n)/\epsilon_d)^d)$. By

setting d to be $\Theta(\log\log n)$, one can see that the second factor in the product is $n^{o(1)}$ and the first factor is an expression that tends to $n^{4/5}$. Thus, the overall complexity is $O(n^{\frac{4}{5}+o(1)})$. □

6 Lower bounds for $(1,3,2)$-freeness testers on hypergrids

In this section, we prove the following lower bounds for nonadaptive and adaptive ϵ-testers for $(1,3,2)$-freeness on the hypergrid $[n]^2$.

Theorem 6.1. *Any one-sided error nonadaptive ϵ-tester for $(1,3,2)$-freeness on $[n]^2$ has query complexity $\Omega(n)$, for every $\epsilon \leq 1/81$.*

The existence of a $o(n)$-query *adaptive* tester for $(1,3,2)$-freeness (Theorem 1.4) shows that adaptivity does indeed lead to more efficient testers. On the other hand, the following theorem states that even with adaptivity, testing $(1,3,2)$-freeness cannot be done much more efficiently.

Theorem 6.2. *Any one-sided error adaptive ϵ-tester for $(1,3,2)$-freeness on $[n]^2$ has query complexity $\Omega(\sqrt{n})$, for every $\epsilon \leq 4/81$.*

To prove Theorem 6.1 and Theorem 6.2, the high level idea is to follow [NRRS19], in which they prove an $\Omega(\sqrt{n})$-query lower bound for *nonadaptively* testing $(1,3,2)$-freeness on the line $[n]$. Namely, we reduce an "intersection-search" problem (Theorem 6.3) for two 2-*dimensional* $3n \times 3n$ 'monotone arrays' (w.r.t. $\prec$), to the problem of testing $(1,3,2)$-freeness on the 2-dimensional hypergrid $[9n]^2$. This reduction (Theorem 6.6) is blackbox deterministic, and as such, any lower bound for it (against nonadaptive/adaptive randomized algorithms), implies a corresponding lower bound for testing $(1,3,2)$-freeness. However, unlike in [NRRS19], where the corresponding 1-dimensional problem could be solved adaptively in $O(\log n)$ queries, we show that our 2-dimensional variant requires $\Omega(\sqrt{n})$ adaptive queries. This is shown by observing that Theorem 6.3 'embeds' in it $\Omega(n)$ independent instances of a variant of the "birthday problem" (Theorem 6.4), each of which are to be solved on unstructured sets of size $\Omega(n)$.

Problem 6.3 (Intersection-search for two 2-dimensional arrays)**.** *The input to this problem consists of two $3n \times 3n$ arrays A and B, and a constant γ with $0 \leq \gamma \leq 1$ (which we call the intersection parameter). A and B each consist of $9n^2$ distinct integers sorted in ascending order, with respect to the grid (partial) order $\prec$. It is promised that $\left|\{A[\mathbf{i}] \colon \mathbf{i} \in [3n]^2\} \cap \{B[\mathbf{i}] \colon \mathbf{i} \in [3n]^2\}\right| \geq \gamma \cdot (9n^2)$. The goal is to find $\mathbf{i}, \mathbf{j} \in [3n]^2$ such that $A[\mathbf{i}] = B[\mathbf{j}]$.*

Problem 6.4 (Birthday-search)**.** *The input is a pair of m-sized arrays A, B, each of which contain the numbers $[m]$, and are permuted according to some* unknown *permutations π_A, π_B respectively. That is, the input is the pair of permutations π_A and π_B. The goal is to find a pair of indices (i,j) such that $A[i] = B[j]$.*

It is quite clear that birthday-search on m-size arrays can be solved (with high probability) nonadaptively by querying $O(\sqrt{m})$ random indices in each array. It is folklore that adaptivity does not help in this case.

Claim 6.5. *Any adaptive algorithm for birthday-search on m-sized arrays requires $\Omega(\sqrt{m})$ queries to achieve a success probability of $2/3$.* □

The intersection-search problem can be reduced to the problem of testing $(1,3,2)$-freeness of real-valued functions on 2-dimensional hypergrids. We will now formally state this reduction, and defer its proof to the appendix (Theorem C.4).

Lemma 6.6. *Let (A, B, γ) be an input instance of Theorem 6.3. Then, there exists a function $f\colon [9n]^2 \to \mathbb{R}$ that is at least $\gamma/9$-far from $(1,3,2)$-freeness, and there is a one-to-one correspondence between valid solutions of Theorem 6.3, and $(1,3,2)$-appearances in f.*

Observe that Theorem 6.3 can be solved by a simple nonadaptive randomized algorithm that makes a total of $O(n)$ queries to the arrays. One can simply query $O(n)$ indices in A and $O(n)$ indices in B uniformly and independently at random. With probability $\Theta(1)$, this will ensure that we sampled $\mathbf{i}, \mathbf{j} \in [3n]^2$, such that $A[\mathbf{i}] = B[\mathbf{j}]$, provided $\gamma = \Theta(1)$. This is nothing but a variant of the birthday problem. We will prove that this bound is tight and then directly conclude Theorem 6.1, thanks to the reduction Theorem 6.6.

Lemma 6.7. *Let $\mathcal{A}$ be any nonadaptive randomized algorithm for the intersection-search problem Theorem 6.3, with intersection parameter $\gamma \geq 1/9$. If $\mathcal{A}$ makes q queries and $q < n^2/2$, then the success probability of $\mathcal{A}$ is at most $q^2/(4n^2)$.*

The proof of Theorem 6.7 is deferred to the appendix (Theorem C.5). Theorem 6.7 implies that any nonadaptive randomized algorithm must query $\Omega(n)$ indices from A and B so as to solve Theorem 6.3 with probability $\Theta(1)$. The $\Omega(n)$-query lower bound Theorem 6.1 directly follows from Theorem 6.6 and Theorem 6.7. □

The adaptive lower bound Theorem 6.2 is obtained by first proving an $\Omega(\sqrt{n})$-query lower bound for *adaptive* randomized algorithms for Theorem 6.3 that succeed with probability $\Theta(1)$. We state this formally below and defer its proof to the appendix (Theorem C.6).

Lemma 6.8. *Let $\mathcal{A}$ be any adaptive randomized algorithm for the intersection-search problem Theorem 6.3, with intersection parameter $\gamma \geq 4/9$. If $\mathcal{A}$ makes q queries, then the success probability of $\mathcal{A}$ is at most $q^2/(4n+4)$.*

Hence, any adaptive randomized algorithm must query $\Omega(\sqrt{n})$ indices from A and B so as to solve Theorem 6.3 with probability $\Theta(1)$. The $\Omega(\sqrt{n})$-query lower bound Theorem 6.2 directly follows from Theorem 6.6 and Theorem 6.8. □

References

[AAAH01] Michael H. Albert, Robert E. L. Aldred, Mike D. Atkinson, and Derek A. Holton. Algorithms for pattern involvement in permutations. In *Proc.*, volume 2223 of *Lecture Notes in Computer Science*, pages 355–366. Springer, 2001. 2

[AF00] Noga Alon and Ehud Friedgut. On the number of permutations avoiding a given pattern. *J. Comb. Theory A*, 89(1):133–140, 2000. 2

[AR08] Shlomo Ahal and Yuri Rabinovich. On complexity of the subpattern problem. *SIAM J. Discret. Math.*, 22(2):629–649, 2008. 2

[Arr99] Richard Arratia. On the Stanley-Wilf conjecture for the number of permutations avoiding a given pattern. *Electron. J. Comb.*, 6, 1999. 2

[Ban16] Christoph Bandt. Permutation entropy and order patterns in long time series. In *Time Series Analysis and Forecasting*, pages 61–73. Springer International Publishing, 2016. 2

[BC18] Omri Ben-Eliezer and Clément L. Canonne. Improved bounds for testing forbidden order patterns. In *Proc. Twenty-Ninth ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2093–2112. SIAM, 2018. 2, 3, 4, 6

[BCLW19] Omri Ben-Eliezer, Clément L. Canonne, Shoham Letzter, and Erik Waingarten. Finding monotone patterns in sublinear time. In *60th IEEE Symposium on Foundations of Computer Science, FOCS 2019*, pages 1469–1494. IEEE Computer Society, 2019. 2, 3, 4, 6, 7

[BCS18] Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. A $o(d) \cdot \text{polylog } n$ monotonicity tester for Boolean functions over the hypergrid $[n]^d$. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2133–2151. SIAM, 2018. 6

[BCS20] Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. Domain reduction for monotonicity testing: A $o(d)$ tester for Boolean functions in d-dimensions. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 1975–1994. SIAM, 2020. 6

[BCS23a] Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. A $\mathrm{d}^{1/2+\mathrm{o}(1)}$ monotonicity tester for Boolean functions on d-dimensional hypergrids. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 1796–1821. IEEE, 2023. 6

[BCS23b] Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. Directed isoperimetric theorems for Boolean functions on the hypergrid and an $\tilde{\mathrm{o}}(\mathrm{n}\sqrt{\mathrm{d}})$ monotonicity tester. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023*, pages 233–241. ACM, 2023. 6

[Bel18] Aleksandrs Belovs. Adaptive lower bound for testing monotonicity on the line. In *Approximation*, pages 31:1–31:10, 2018. 2

[BGJ+12] Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012. 2

[BKKM23] Mark Braverman, Subhash Khot, Guy Kindler, and Dor Minzer. Improved monotonicity testers via hypercube embeddings. In *14th Innovations in Theoretical Computer Science Conference, ITCS 2023*, volume 251 of *LIPIcs*, pages 25:1–25:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. 6

[BKM21] Benjamin Aram Berendsohn, László Kozma, and Dániel Marx. Finding and counting permutations via CSPs. *Algorithmica*, pages 1–26, 2021. 2

[BKO24] Benjamin Aram Berendsohn, László Kozma, and Michal Opler. Optimization with pattern-avoiding input. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 671–682. ACM, 2024. 2

[BKR23] Hadley Black, Iden Kalemaj, and Sofya Raskhodnikova. Isoperimetric inequalities for real-valued functions with applications to monotonicity testing. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023*, volume 261 of *LIPIcs*, pages 25:1–25:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. 6

[BKSJ19] Sebastian Berger, Andrii Kravtsiv, Gerhard Schneider, and Denis Jordan. Teaching ordinal patterns to a computer: Efficient encoding algorithms based on the Lehmer code. *Entropy*, 21(10), 2019. 2

[BLW22] Omri Ben-Eliezer, Shoham Letzter, and Erik Waingarten. Finding monotone patterns in sublinear time, adaptively. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022*, volume 229 of *LIPIcs*, pages 17:1–17:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. 2, 3, 6, 7

[Bón97] Miklós Bóna. *Permutations with restrictions and associated structures.* Ph. D. thesis, Massachusetts Institute of Technology, 1997. Available at MIT DSpace. 2

[Bón99] Miklós Bóna. The solution of a conjecture of Stanley and Wilf for all layered patterns. *J. Comb. Theory A*, 85(1):96–104, 1999. 2

[BP02] Christoph Bandt and Bernd Pompe. Permutation entropy: A natural complexity measure for time series. *Phys. Rev. Lett.*, 88:174102, 2002. 2

[CDJS17] Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. Property testing on product distributions: Optimal testers for bounded derivative properties. *ACM Trans. Algorithms*, 13(2):20:1–20:30, 2017. 2

[CDST15] Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost) $n^{\frac{1}{2}}$ non-adaptive queries. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 519–528. ACM, 2015. 6

[CGJ+23] Parinya Chalermsook, Manoj Gupta, Wanchote Jiamjitrak, Nidia Obscura Acosta, Akash Pareek, and Sorrachai Yingchareonthawornchai. Improved pattern-avoidance bounds for greedy BSTs via matrix decomposition. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 509–534. SIAM, 2023. 2

[CGK+15] Parinya Chalermsook, Mayank Goswami, László Kozma, Kurt Mehlhorn, and Thatchaphol Saranurak. Pattern-avoiding access in binary search trees. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 410–423. IEEE Computer Society, 2015. 2

[CPY24] Parinya Chalermsook, Seth Pettie, and Sorrachai Yingchareonthawornchai. Sorting pattern-avoiding permutations via 0-1 matrices forbidding product patterns. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 133–149. SIAM, 2024. 2

[CS13] Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proc. ACM Symposium on Theory of Computing (STOC) 2013*, pages 419–428, 2013. 2, 6

[CS14] Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory Comput.*, 10:453–464, 2014. 2, 4, 6

[CS16] Deeparnab Chakrabarty and C. Seshadhri. An o(n) monotonicity tester for Boolean functions over the hypercube. *SIAM J. Comput.*, 45(2):461–472, 2016. 6

[CST14] Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 286–295. IEEE Computer Society, 2014. 6

[CWX17] Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond Talagrand functions: New lower bounds for testing monotonicity and unateness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 523–536. ACM, 2017. 6

[DGL+99] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *Randomization, Approximation, and Combinatorial Algorithms and Techniques, Third International Workshop on Randomization and Approximation Techniques in Computer Science, and Second International Workshop on Approximation Algorithms for Combinatorial Optimization Problems RANDOM-APPROX'99*, volume 1671 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 1999. 2, 6

[DRTV18] Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin Varma. Erasure-resilient property testing. *SIAM J. Comput.*, 47(2):295–329, 2018. 3, 5

[EKK+00] Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000. 2, 6

[Fis04] Eldar Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1):107–116, 2004. 2, 3, 4, 6

[FLN+02] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In John H. Reif, editor, *Proc. on 34th ACM Symposium on Theory of Computing (STOC 2002)*, pages 474–483. ACM, 2002. 6

[FN07] Eldar Fischer and Ilan Newman. Testing of matrix-poset properties. *Comb.*, 27(3):293–327, 2007. 6

[Fox13] Jacob Fox. Stanley-Wilf limits are typically exponential, 2013. 2

[GGL+00] Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Comb.*, 20(3):301–337, 2000. 2, 6

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. 2

[GM14] Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. In *Proc. Twenty-Fifth ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 82–101. SIAM, 2014. 2

[JNdM21] Y. Jang, J. Nešetřil, and P. Ossona de Mendez. Füredi-Hajnal and Stanley-Wilf conjectures in higher dimensions, 2021. 8, 19

[KL03] Karsten Keller and Heinz Lauffer. Symbolic analysis of high-dimensional time series. *Int. J. Bifurc. Chaos*, 13(9):2657–2668, 2003. 2

[Kla00] Martin Klazar. The Füredi-Hajnal conjecture implies the Stanley-Wilf conjecture. In *Formal Power Series and Algebraic Combinatorics*, pages 250–255, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. 2

[KMS18] Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and Boolean isoperimetric-type theorems. *SIAM J. Comput.*, 47(6):2238–2276, 2018. 6

[Knu68] Donald E. Knuth. *The art of computer programming, volume I: Fundamental algorithms*. Addison-Wesley, 1968. 1

[MT04] Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *J. Comb. Theory, Ser. A*, 107(1):153–160, 2004. 2, 8, 19

[NRRS19] Ilan Newman, Yuri Rabinovich, Deepak Rajendraprasad, and Christian Sohler. Testing for forbidden order patterns in an array. *Random Struct. Algorithms*, 55(2):402–426, 2019. 2, 3, 4, 6, 7, 8, 9, 15, 25, 38, 42

[NV21] Ilan Newman and Nithin Varma. New sublinear algorithms and lower bounds for LIS estimation. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 100:1–100:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 10, 45

[NV24] Ilan Newman and Nithin Varma. Strongly sublinear algorithms for testing pattern freeness. *TheoretiCS*, 3, 2024. 2, 3, 5, 6, 7, 8, 9, 18, 19, 45

[PR21] Arthur A. B. Pessa and Haroldo V. Ribeiro. Ordpy: A Python package for data analysis with permutation entropy and ordinal network methods. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(6):063110, 2021. 2

[PRV18] Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Nithin Varma. Parameterized property testing of functions. *ACM Trans. Comput. Theory*, 9(4):17:1–17:19, 2018. 6

[PRW22] Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of Boolean functions. *Random Struct. Algorithms*, 60(2):233–260, 2022. 5

[RLM+07] Osvaldo A Rosso, HA Larrondo, María Teresa Martin, A Plastino, and Miguel A Fuentes. Distinguishing noise from chaos. *Physical review letters*, 99(15):154102, 2007. 2

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. 2

[RZL+12] Haroldo V. Ribeiro, Luciano Zunino, Ervin K. Lenzi, Perseu A. Santoro, and Renio S. Mendes. Complexity-entropy causality plane as a complexity measure for two-dimensional patterns. *PLOS ONE*, 7(8):1–9, 2012. 2

[Yao77] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 222–227. IEEE Computer Society, 1977. 43

[ZSR12] L. Zunino, M. C. Soriano, and O. A. Rosso. Distinguishing chaotic and stochastic dynamics from time series by using a multiscale symbolic approach. *Phys. Rev. E*, 86:046210, 2012. 2

[ZZRP12] Massimiliano Zanin, Luciano Zunino, Osvaldo A Rosso, and David Papo. Permutation entropy and its main biomedical and econophysics applications: A review. *Entropy*, 14(8):1553–1577, 2012. 2

A Erasure-resilient monotonicity testing over high dimensions

In this section, we describe an ER monotonicity tester for real-valued functions over hypergrids of arbitrary dimension. Such a tester for 2-dimensional hypergrids is a key subroutine in our pattern freeness testers in Section 4 and Section 5.

Theorem A.1. *There is a one-sided error δ-erasure-resilient ϵ-tester for monotonicity of functions $f\colon [n]^2 \to \mathbb{R}$ making $\widetilde{O}\left(\frac{\log^2(1/\epsilon)}{\epsilon(1-\delta)}\right)$ queries.*

A.1 A simple tester

We first describe a simple one-sided error erasure-resilient monotonicity tester for real-valued functions over the 2-dimensional hypergrid $[n]^2$. For convenience, we describe a $(1,2)$-freeness tester, which can be easily modified to be a monotonicity (i.e., $(2,1)$-freeness) tester. Specifically, we prove the following theorem and later discuss the additional ideas needed to show Theorem A.1.

Theorem A.2. *There is a one-sided error δ-erasure-resilient ϵ-tester for $(1,2)$-pattern freeness of functions $f\colon [n]^2 \to \mathbb{R}$ with query complexity $\widetilde{O}\left(\frac{1}{\epsilon^2(1-\delta)}\right)$ for all $\epsilon, \delta \in (0,1)$.*

Let $f\colon [n]^2 \to \mathbb{R}$. For simplicity, let us initially assume that f has *no* erased points. Since our goal is to design a one-sided error tester, we may also assume that f is ϵ-far from being $(1,2)$-free. The tester we describe rejects upon finding a $(1,2)$-appearance. By Theorem 2.5, there exists a matching M of $(1,2)$-appearances of size $|M| \geq \epsilon n^2/2$. Throughout this section, fix such a matching M.

Given a tuple (p,q) with $p,q \in [n]^2$ and $p \prec q$, we place (p,q) in the *bucket* (α,β), where $\alpha,\beta \in \{0,1,\ldots,\log_2 n\}$ in the following manner – at level i, partition $[n]$ in 2^i equal parts in the natural way (for instance, at level 2, the parts are $[1,n/4]$, $(n/4,n/2]$, $(n/2,3n/4]$, and $(3n/4,n]$), and stop at the lowest level α in which the x-coordinates $x(p)$ and $x(q)$ of p and q fall into different parts. That is, the natural partition of the array into 2^α parts is the *coarsest* partition that separates $x(p)$ and $x(q)$. Note that since $p \prec q$, this would ensure that $x(q)$ belongs to the part of $[n]$ immediately to the right of the part which contains $x(p)$, provided $x(p) < x(q)$. If $x(p) = x(q)$, we set $\alpha := 0$. Similarly, let β be the lowest level that separates the y-coordinates $y(p)$ and $y(q)$ of p and q. Then, we say that the tuple (p,q) belongs to the bucket (α,β). Denoting $\ell = \log_2 n + 1$, we see that every such tuple belongs to one of ℓ^2 buckets.

Then we define

$$M_{\alpha,\beta} := \{(p,q) \in M\colon (p,q) \text{ belongs to the bucket } (\alpha,\beta)\}.$$

The sets $M_{\alpha,\beta}$ partition M into $O(\log^2 n)$ submatchings. An immediate conclusion is that for some $\alpha^*,\beta^* \in \{0,1,\ldots,\log_2 n\}$, $|M_{\alpha^*,\beta^*}| \geq \epsilon n^2/2\ell^2$, i.e., $|M_{\alpha^*,\beta^*}| = \Omega(\epsilon n^2/\log^2 n)$.

Indeed, we do not have prior knowledge of such M, α^*, and β^*. Therefore, we first design a tester for a specified pair of values α and β within the range $\mathcal{S} := \{0,1,\ldots,\log_2 n\}$ and a proximity parameter $\gamma \in (0,1)$. Our final tester is obtained by running this tester for all possible values of $\alpha,\beta \in \mathcal{S}$ and for the specific proximity parameter $\Omega(\epsilon/\log^2 n)$ (see Theorem A.2). The erasure-resilience of the tester comes easily at a cost of $O(1/(1-\delta))$, where δ is an upper bound on the fraction of erasures in the domain.

The basic test $T(\alpha, \beta, \gamma)$ given $\alpha, \beta \in \mathcal{S}$ and $\gamma \in (0,1)$. Let $\alpha, \beta \in \mathcal{S}$. We partition the domain $[n]^2$ into *boxes* of side lengths $n/2^\alpha$ and $n/2^\beta$ along the x-axis and the y-axis respectively, in the natural way, when $\alpha, \beta \geq 1$.[9] Let $\mathcal{B}$ be this set of boxes. For each box $B \in \mathcal{B}$, let $x_{\max}(B)$ and $y_{\max}(B)$ denote the maximum x-coordinate and y-coordinate values of B respectively. We define the *diagonally adjacent box* B' of B as follows:[10]

$$B' := \left\{ p \in [n]^2 \colon x_{\max}(B) < x(p) \leq x_{\max}(B) + n/2^\alpha, \text{ and } y_{\max}(B) < y(p) \leq y_{\max}(B) + n/2^\beta \right\}$$

For each edge $(p,q) \in M_{\alpha,\beta}$ with $p \in B$, it follows that $q \in B'$. Let $M(B, B')$ denote the set of all $(p,q) \in M_{\alpha,\beta}$ with $p \in B$. Then, $M_{\alpha,\beta} = \bigcup_{B \in \mathcal{B}} M(B, B')$. Further, for any pair of distinct boxes $B_1, B_2 \in \mathcal{B}$, the sets $M(B_1, B_1')$ and $M(B_2, B_2')$ are disjoint and so, the sets $M(B, B')$ form a partition of $M_{\alpha,\beta}$.

The test $T(\alpha, \beta, \gamma)$ is as follows:

1. Sample $B \in \mathcal{B}$ uniformly at random.
2. TEST(B, B', γ): Sample $4/\gamma$ points uniformly at random from B and $4/\gamma$ points uniformly at random from B'. Then query f at all of these $8/\gamma$ points and **reject** if a $(1,2)$-appearance is found among these queried points.

We first prove the following claim about the procedure TEST(B, B', γ), which is Step 2 of test $T(\alpha, \beta, \gamma)$.

Claim A.3. *If $M(B, B') \geq \gamma \cdot |B|$, then* TEST$(B, B', \gamma)$ *finds a $(1,2)$-appearance in f with probability $\Theta(1)$. Its query complexity is $O(1/\gamma)$.*

Proof. The query complexity is clear from the description of TEST(B, B', γ). Let the 1-legs of edges in $M(B, B')$ be $p_1, \ldots, p_k$ where $k \geq \gamma \cdot |B|$ and $f(p_1) \leq \cdots \leq f(p_k)$. Let Q denote the set of corresponding 2-legs (in $M(B, B')$) to these 1-legs. Recall that the points in Q are all in B'. Let $m := (k+1)/2$. Then there are at least m indices i such that $f(p_i) \leq f(p_m)$ and at least m indices $q \in Q$ for which $f(p_m) \leq f(q)$ (those that correspond to 1-legs $p_i,\ i > m$). Let Q_H contain these q's. Then, any pair $\{(p_i, q) \colon\ i \leq m,\ q \in Q_H\}$ forms a $(1,2)$-appearance. Thus, the probability that our sample of $4/\gamma$ points from B intersects $\{p_i,\ i \leq m\}$ is at least $1 - (1 - \frac{\gamma}{2})^{4/\gamma} \geq 1 - e^{-2}$, and similarly, the same holds for the probability that we sample a point in Q_H. Altogether, with $\Theta(1)$ probability, we succeed in finding a $(1,2)$-appearance. ☐

Let $\alpha^*, \beta^* \in \mathcal{S}$ be such that $|M_{\alpha^*,\beta^*}| \geq \epsilon n^2/2\ell^2$. We let $\epsilon_1 := \epsilon/\ell^2$. We will now show that $T(\alpha^*, \beta^*, \epsilon_1/4)$ finds a $(1,2)$-appearance in f with probability $\Theta(\epsilon_1)$.

Lemma A.4. *The algorithm $T(\alpha^*, \beta^*, \epsilon_1/4)$ finds a $(1,2)$-appearance in f with probability $\Theta(\epsilon_1)$. Its query complexity is $O(1/\epsilon_1)$.*

[9] If $\alpha = 0$, then the side length of each box on the x-axis is set to 0. The case $\beta = 0$ is taken care of similarly. Note that at least one of α and β is nonzero, as we only place tuples (p,q) with $p \neq q$ in our buckets.

[10] If $\alpha = 0$, we set $x_{\max}(B') := x_{\max}(B) = x(p)$. The case $\beta = 0$ is taken care of similarly.

Proof. The query complexity is clear from the description of $T(\alpha, \beta, \gamma)$. Since $|M_{\alpha^*,\beta^*}| \geq \epsilon_1 n^2/2$ and $|M_{\alpha^*,\beta^*}| = \sum_{B \in \mathcal{B}} |M(B, B')|$, we have $\mathbb{E}_{B \in \mathcal{B}}[|M(B, B')|] \geq \epsilon_1 |B|/2$. Hence, by the reverse Markov inequality, the tester samples a box B for which $|M(B, B')| \geq \epsilon_1 |B|/4$ with probability at least $\epsilon_1/4$. Conditioning on the event that B indeed has this many appearances, Theorem A.3 implies that the success probability in finding a violation is $\Theta(1)$. Hence, $T(\alpha^*, \beta^*, \epsilon_1/4)$ finds a $(1, 2)$-appearance in f with probability $\Theta(\epsilon_1)$. □

We will now describe an erasure-resilient tester for $(1, 2)$-freeness based on Theorem A.4.

Proof of Theorem A.2. Let ϵ_1 as defined above. Our tester runs $T(\alpha, \beta, \epsilon_1/4)$, independently $\Theta\left(\frac{1}{\epsilon_1} \log \log n\right)$ times, for each $\alpha, \beta \in \mathcal{S}$, to have a $\Theta(1)$ success probability. Every time we need to query a point and it happens to be erased, we repeat the process by choosing another point independently. So in expectation, we invest $O(1/(1-\delta))$ queries in order to query a single point, and this reduces the test to the case where no point is erased. Hence the tester makes a total of $O\left(\frac{\log^6 n \log \log n}{\epsilon^2(1-\delta)}\right)$ queries. □

A.2 Testing monotonicity of partial functions over $[n]^2$ and improving dependence on ϵ

Theorem A.2 asserts the correctness and complexity of an erasure-resilient monotonicity tester, which serves our purpose for $(1, 2, 3)$-freeness testing. However, for testing $(1, 3, 2)$-freeness, we need the ability to test monotonicity of partial functions $f \colon P \to \mathbb{R}$, where P is an arbitrary subset of $[n]^2$. The subdomain P is known to the tester. An erasure-resilient monotonicity tester as described above can be used for partial function monotonicity testing. However, P can be arbitrarily small in many cases and the inverse dependence of the query complexity on $|P|$ (from Theorem A.2) is not desirable. To address this issue, we design an alternate tester whose query complexity does not depend on $|P|$. Furthermore, we improve upon the quadratic dependence of the query complexity on $1/\epsilon$. We stress that the tester crucially depends on the fact that P is known (unlike in the erasure-resilient case).

Theorem A.5. *There is a one-sided error ϵ-tester for $(1, 2)$-pattern freeness of partial functions $f \colon P \to \mathbb{R}$, where $P \subseteq [n]^2$, with query complexity $\widetilde{O}\left(\frac{\log^2(1/\epsilon)}{\epsilon}\right)$.*

Recall that f has a matching M of violations of size at least $\epsilon|P|/2$. Just as in Appendix A.1, let n be a power of 2, $\ell := \log_2 n + 1$, $\mathcal{S} := \{0, 1, \ldots, \log_2 n\}$, $\alpha, \beta \in \mathcal{S}$, $\epsilon_1 := \epsilon/\ell^2$, so that there exists a pair (α^*, β^*) and a corresponding submatching M_{α^*,β^*} of size $|M_{\alpha^*,\beta^*}| \geq \epsilon_1 |P|/2$. Further, we partition the domain $[n]^2$ (regardless of P) into boxes of appropriate sizes, as in Appendix A.1, and define the *adjacent regions* of boxes exactly as we did there. This implies that all the 1- and 2-legs of each $(1, 2)$-appearance in M_{α^*,β^*} are in a box $B \in \mathcal{B}$ and the corresponding diagonally adjacent box B' (see Appendix A.1) respectively. As in Appendix A.1, we only describe a basic tester $\text{TEST}^*(\alpha, \beta)$ for the scales α, β.

We note that we did not make any assumptions on P so far, and the partition of the domain into boxes is oblivious of P. In particular, some of the boxes may not contain any points of P. For a box B, let $P(B) := P \cap B$. We define the following natural probability distribution $\mathcal{D}_P$ on the set $\mathcal{B}$ of boxes by $\Pr_{\mathcal{D}_P}(B) := \frac{|P(B)|}{|P|}$.

The improved algorithm $\mathrm{TEST}^*(\alpha, \beta)$**.**

1. Let $\ell := \log_2 n + 1, \epsilon_1 := \frac{\epsilon}{\ell^2}, r := \log_2\left(\frac{4}{\epsilon_1}\right)$.
2. For each $k \in [2r]$, perform the following steps:
   (a) Sample $r_k := \frac{20r}{2^k \epsilon_1}$ boxes $B_1, \ldots, B_{r_k}$ from $\mathcal{B}$ independently according to the distribution $\mathcal{D}_P$.
   (b) For each pair (B_i, B_i'), sample $q_k := 4 \cdot 2^k$ points from $B_i \cap P$ and q_k points from $B_i' \cap P$ independently and uniformly at random, and query f at all of these points. If a violation to monotonicity is found, we **reject**.

Just as in Appendix A.1, we fix a matching M of violations and $\alpha^*, \beta^* \in \mathcal{S}$ for which $|M| \geq \frac{\epsilon|P|}{2}$ and $|M_{\alpha^*,\beta^*}| \geq \frac{\epsilon|P|}{2\ell^2}$. We will now show that $\mathrm{TEST}^*(\alpha^*, \beta^*)$ finds a violation with high probability, and compute its query complexity.

Claim A.6. *The algorithm* $\mathrm{TEST}^*(\alpha^*, \beta^*)$ *finds a violation with probability* $\Theta(1)$*. Its query complexity is* $O\left(\frac{\log^2\left(\frac{1}{\epsilon_1}\right)}{\epsilon_1}\right)$.

Proof. The total number of queries is bounded by $\sum_{k \in [2r]} r_k \cdot (2q_k) = (2r) \cdot \frac{20r}{\epsilon_1} \cdot 8$, so that the query complexity is $O\left(\frac{\log^2\left(\frac{1}{\epsilon_1}\right)}{\epsilon_1}\right)$ as claimed.

To estimate the success probability, for each $B \in \mathcal{B}$, we denote by $\epsilon_B := \frac{|M(B,B')|}{|P(B)|}$ the normalized number of violations from M_{α^*,β^*} with their 1-leg in B and 2-leg in B'. Then

$$\mathop{\mathbb{E}}_{B \in \mathcal{B}}[\epsilon_B] = \sum_{B \in \mathcal{B}} \Pr_{\mathcal{D}_P}(B) \cdot \epsilon_B = \sum_{B \in \mathcal{B}} \frac{|P(B)|}{|P|} \cdot \frac{|M(B,B')|}{|P(B)|} \geq \frac{\epsilon_1}{2}, \tag{A.1}$$

where the expectation is taken with respect to the distribution $\mathcal{D}_P$ on $\mathcal{B}$.

Now, for each $k \in [2r]$, let p_k be the probability that B (chosen according to $\mathcal{D}_P$) has $2^{-k} < \epsilon_B \leq 2^{-k+1}$.

Assume that for some fixed k, we have $p_k \geq \frac{2^k \epsilon_1}{16r}$. We will bound the success probability by the event that the algorithm will succeed in iteration k. Indeed, at iteration k, the algorithm will pick a box B with $\epsilon_B > 2^{-k}$ with probability at least p_k. Since $r_k \geq \frac{1}{p_k}$ by our assumption on p_k, with probability $\Theta(1)$, at least one of the sampled boxes B in the k^{th} iteration will have $\epsilon_B > 2^{-k}$. Conditioned on this, Theorem A.3 (with $\gamma = 2^{-k}$) then implies that for such a B, picking q_k points each from B and B' uniformly at random and querying them will ensure that we find a violation with probability $\Theta(1)$. Altogether, this proves that the success probability is $\Theta(1)$ under the above assumption on p_k.

We are now left with the case that, for every $k \in [2r]$, $p_k < \frac{2^k \epsilon_1}{16r}$. In this case, we have

$$\mathop{\mathbb{E}}_{B \in \mathcal{B}}[\epsilon_B] \leq \sum_{k \in [2r]} p_k \cdot 2^{-k+1} + 1 \cdot 2^{-2\log\left(\frac{4}{\epsilon_1}\right)}$$

where the last term accounts for an upper bound on the expected value of ϵ_B over all the boxes for which $\epsilon_B < 2^{-2\log\left(\frac{4}{\epsilon_1}\right)}$.

Simplifying the bound above gives us $\mathop{\mathbb{E}}_{B\in\mathcal{B}}[\epsilon_B] \leq 2\cdot 2r\cdot\frac{\epsilon_1}{16r}+\left(\frac{\epsilon_1}{4}\right)^2 < \frac{\epsilon_1}{2}$, contradicting Equation (A.1). This completes the proof. □

Theorem A.6 directly implies Theorem A.5, and can be easily adapted and used in place of Theorem A.4 to get Theorem A.1, which is an improved version of Theorem A.2.

Proof of Theorem A.1. We can obtain a basic erasure-resilient monotonicity tester ER-TEST$^*(\alpha,\beta)$ corresponding ot each bucket (α,β) by only making a couple of modifications to the basic tester TEST$^*(\alpha,\beta)$ for partial functions given in the proof of Theorem A.5. In particular, (i) in Step 2a, the boxes $B_1,\ldots,B_{r_k}$ should be sampled independently and *uniformly at random* (as opposed to sampling according to the distribution $\mathcal{D}_P$ in the case of partial functions, since we do not have access to the list of nonerased points apriori), and (ii) the distance parameter ϵ_1 must be set to $\frac{\epsilon\cdot(1-\delta)}{\ell^2}$, where $\delta\in(0,1)$ is an upper bound on the fraction of erased points in the domain $[n]^2$, and $\ell := \log_2 n+1$. Then, as was done in the case of partial functions, the basic tester ER-TEST$^*(\alpha,\beta)$ is run independently for each of the ℓ^2 buckets (α,β), where $\alpha,\beta\in\{0,1,\ldots,\log_2 n\}$.

In this case, observe that $\mathop{\mathbb{E}}_{B\in_R\mathcal{B}}[\epsilon_B] \geq \frac{\epsilon_1}{2}$, where $\epsilon_B := \frac{|M(B,B')|}{|B|}$ denotes the normalized distance parameter, $\epsilon_1 := \frac{\epsilon\cdot(1-\delta)}{\ell^2}$, and the expectation is taken with respect to the uniform distribution on the set of boxes $\mathcal{B}$. The rest of the proof is analogous to Theorem A.6. Therefore, we have a δ-ER ϵ-tester for monotonicity of real-valued functions on $[n]^2$ with query complexity $\ell^2\cdot O\left(\frac{\log^2\left(\frac{1}{\epsilon_1}\right)}{\epsilon_1}\right)$, which is $\widetilde{O}\left(\frac{\log^2(1/\epsilon)}{\epsilon(1-\delta)}\right)$. □

For using the monotonicity testers described above in 3-pattern testing, we require more. We need the testers to output not just an arbitrary violation to monotonicity, but rather a "typical" one as we remark below.

Remark A.7. Let M be a matching of $(1,2)$-pairs on a domain D. Theorem A.1 describes a tester for monotonicity that finds a violation (p_1,p_2) with p_1 and p_2 being the 1-leg and the 2-leg of some pairs in M (but not necessarily a pair by itself) with constant probability. The tester induces a probability distribution P_M on the 1- and 2-legs of the pair it produces and in particular, a distribution on the 1-leg p_1. It is easy to see that this probability is uniform over the 1-legs of pairs in M.

Remark A.8. Our monotonicity testers work even when the input function is over a rectangular grid $[n_x]\times[n_y]$, and the dependence of the query complexity on the input size is then polylog$(\max\{n_x,n_y\})$.

A.3 Generalizing to any dimension

The ideas described in Appendix A.1 and Appendix A.2 to construct partial function and ER monotonicity testers for functions $f\colon[n]^2\to\mathbb{R}$ generalize in a natural way when the domain is $[n]^d$, for any $d\geq 1$. We simply extend the notion of bucketing described in Appendix A.1 to pairs of points in $[n]^d$. To this end, given a tuple (p,q) with $p,q\in[n]^d$ and $p\prec q$, we place (p,q) in the bucket $(\ell_1,\ldots,\ell_d)$ where $\ell_1,\ldots,\ell_d\in\{0,1,\ldots,\log_2 n\}$ if, for each $i\in[d]$, the coarsest partition of $[n]$ that separates $p[i]$ and $q[i]$ (i.e., the i-coordinates of p and q respectively) is constructed at level

ℓ_i (i.e., when $[n]$ is partitioned into 2^{ℓ_i} equal parts). If $p[i] = q[i]$, we set $\ell_i := 0$. Given this bucketing scheme, the *boxes* into which we partition the domain $[n]^d$ will have side-length $n/2^{\ell_i}$ in the i-coordinate if $\ell_i \geq 1$ and 0 otherwise, for each $i \in [d]$.

Let $f\colon [n]^d \to \mathbb{R}$ be ϵ-far from being $(1,2)$-free. Then, there exists a matching M of $(1,2)$-appearances of size $|M| \geq \epsilon n^d/2$, which we fix. With the bucketing scheme described above, there then exists a submatching M_{ℓ^*} of size $|M_{\ell^*}| \geq \epsilon n^d/2(\log_2 n + 1)^d$ which consists of all the $(1,2)$-appearances in M that belong to the bucket $\ell^* := (\ell_1^*, \ldots, \ell_d^*)$.

We can then obtain a basic monotonicity tester $\text{TEST}^*(\ell_1, \ldots, \ell_d)$ (that corresponds to the bucket $(\ell_1, \ldots, \ell_d)$) for partial functions $f\colon P \to \mathbb{R}$, where $P \subseteq [n]^d$, that is analogous to the algorithm $\text{TEST}^*(\alpha, \beta)$ for $d = 2$ which is given in Appendix A.2. The only modification to be made is to the distance parameter ϵ_1, which should now be set to $\frac{\epsilon}{\ell^d}$, where $\ell := \log_2 n + 1$. The final tester is then simply $\text{TEST}^*(\ell_1, \ldots, \ell_d)$ run independently on each of the ℓ^d bucket types $(\ell_1, \ldots, \ell_d)$, where $\ell_1, \ldots, \ell_d \in \{0, 1, \ldots, \log_2 n\}$.

The final query complexity of this monotonicity tester for real-valued partial functions defined on some given subset P of $[n]^d$ is then $\ell^d \cdot O\left(\frac{\log^2\left(\frac{1}{\epsilon_1}\right)}{\epsilon_1}\right)$, which is $O\left(\frac{\log^{O(d)} n}{\epsilon} \cdot \log^2 \frac{1}{\epsilon}\right)$. The query complexity is independent of $|P|$, as desired. We have proved the following.

Theorem A.9. *There is a one-sided error ϵ-tester for monotonicity of partial functions $f\colon P \to \mathbb{R}$, where P is a (known) subset of $[n]^d$, with query complexity $O\left(\frac{\log^{O(d)} n}{\epsilon} \cdot \log^2 \frac{1}{\epsilon}\right)$, for all $\epsilon \in (0,1)$.*

One can then obtain a basic ER-tester $\text{ER-TEST}^*(\ell_1, \ldots, \ell_d)$ (that corresponds to the bucket $(\ell_1, \ldots, \ell_d)$) for functions $f\colon [n]^d \to \mathbb{R}$, from the tester $\text{TEST}^*(\ell_1, \ldots, \ell_d)$ for partial functions, just like it was done in the proof of Theorem A.1 towards the end of Appendix A.2. Namely, the boxes should be sampled independently and uniformly at random, and the distance parameter ϵ_1 should be set to $\frac{\epsilon \cdot (1-\delta)}{\ell^d}$, where $\delta \in (0,1)$ is an upper bound on the fraction of erased points in the domain $[n]^d$, and $\ell := \log_2 n + 1$. The final query complexity of this δ-ER ϵ-tester for monotonicity of real-valued functions on $[n]^d$ is then $\ell^d \cdot O\left(\frac{\log^2\left(\frac{1}{\epsilon_1}\right)}{\epsilon_1}\right)$, which is $O\left(\frac{\log^{O(d)} n}{\epsilon(1-\delta)} \cdot \log^2 \frac{1}{\epsilon}\right)$. We have proved the following.

Theorem A.10. *There is a δ-erasure-resilient ϵ-tester for monotonicity of functions $f\colon [n]^d \to \mathbb{R}$ with one-sided error, and query complexity $O\left(\frac{\log^{O(d)} n}{\epsilon(1-\delta)} \cdot \log^2 \frac{1}{\epsilon}\right)$, for all $\epsilon, \delta \in (0,1)$.*

B Missing proofs from Section 4

In this section, we provide the pseudocodes and omitted proofs for some of the cases of our $(1,2,3)$-freeness tester outlined in Section 4.

Let $f\colon [n]^2 \to \mathbb{R}$ be ϵ-far from $(1,2,3)$-free. Then, f contains a matching M of $(1,2,3)$-appearances of size $|M| \geq \epsilon n^2/3$. This matching M is partitioned into submatchings $M = M_0 \cup M_1 \cup M_2 \cup M_3$, as explained in Section 4.1.1. The subcase in which M_0 is large, i.e., $|M_0| \geq \epsilon n^2 \cdot \left(1 - \frac{1}{\log n}\right)$ has been described in Section 4.2.1 and Section 4.3. The subcase in which M_3 is large, i.e., $|M_3| \geq \epsilon n^2/(3 \log n)$, has been described in Section 4.2.4. In what follows, we complete the description of the other cases.

B.1 M_1 is large

Let $M_1 \subseteq M$ denote the set of $(1,2,3)$-appearances with all their legs in the same row (or column) of boxes in $G_m^{(2)}$, but not all in the same box.

The matching of $(1,2,3)$-appearances such that the 1- and 2-legs are in the same box, but the 3-leg is in a different box, can be further split into these appearances for which the y-spacing between 2- and 3-legs is more than that between the 1- and 2-legs. We denote this submatching as M_{11}, and denote by M_{12} the submatching for which the y-spacing between the 1- and 2-legs in the $(1,2,3)$-appearance is more than that between the 2- and 3-legs. This leads to two subcases.

B.1.1 M_{11} is large

When M_{11} is large, Algorithm 1 will find a $(1,2,3)$-appearance with probability at least $1-n^{-\Omega(\log n)}$, by making $\widetilde{O}(m/\epsilon^3)$ queries.

Algorithm 1 $(1,2,3)$-freeness testing: M_{11} is large

Repeat $\widetilde{\Theta}(1/\epsilon)$ times:

1. Sample a uniformly random box B. Let R denote the row containing B.
2. For each scale $k \in [\log(n/m)]$,
   (a) Divide R equally into $n/(2^k m)$ horizontal strips.
   (b) Repeat $\widetilde{\Theta}(1/\epsilon)$ times:
      i. Sample and query a uniformly random point p_3 in B.
      ii. Let H' be the strip containing p_3 and let H be the horizontal strip immediately below H'.
      iii. Run a $(1,2)$-freeness tester with proximity parameter $\widetilde{\Theta}(\epsilon/m)$ in the restriction of H to the region of the row R to the left of box B.
3. Return any $(1,2,3)$-appearance from among the points queried.

Suppose that the cardinality of M_{11} is at least $\epsilon' n^2$, where ϵ' is $\epsilon/(c \log n)$ for some absolute constant c. This case is the easy case in which partitioning M_{11} into the exponentially growing buckets according to the y-spacing between the 2- and 3-legs, much like in [NRRS19], allows us to use the 'median' argument. The details follow.

By the assumption on the size of M_{11}, with probability at least $\epsilon'/2$, a uniformly random box has at least $(\epsilon'/2) \cdot (n^2/m^2)$ many 3-legs of appearances in M_{11}. Thus, with probability at least $1 - n^{-\Omega(1)}$, one out of $\widetilde{\Theta}(1/\epsilon)$ sampled boxes contains this number of 3-legs. Let B denote such a box and let R be the row containing B. Let R' be the part of the row R to the left of B.

For each $k \in [\log n]$, we consider equipartitioning R into $n/(2^k m)$ horizontal strips, each being a grid isomorphic to $[n] \times [2^k]$. Let $k' \in [\log n]$ be the scale that contains the largest number of the $(1,2,3)$-appearances in M_{11} whose $1,2$ legs are on a strip and the corresponding 3-leg is in the strip immediately above. That is, this scale contains at least $(\epsilon'/(2\log n)) \cdot (n^2/m^2)$ many 1- and 2-legs, and their corresponding 3-legs, of appearances in M_{11}.

The restriction to B of a uniformly random horizontal strip from this scale contains, in expectation, at least $(\epsilon'/(2\log n)) \cdot (n/m) \cdot 2^{k'}$ many 3-leg appearances. Thus with (high) probability

$p = \Omega(\epsilon'/\log n)$, a sampled strip when restricted to B contains at least $(\epsilon'/(4\log n)) \cdot (n/m) \cdot 2^{k'}$ many 3-leg appearances. Let H denote such a horizontal strip and let H' denote the horizontal strip immediately above H. Sampling a random point from $H' \cap B$ will find a 3-leg from M_{11} with probability $\Omega(p)$. Further, with probability $p/2$ this point will have value higher than the median value of the 3-legs in $H' \cap B$. Assuming the above happens, we only need to find a $(1,2)$-appearance in H, to the left of B and below the sampled point in B that is assumed to be a 3-leg. Now, by the density argument above, this region is $p/(2m\log n)$-far from being $(1,2)$-free. Hence, the $(1,2)$-freeness tester applied on this region will find such corresponding $(1,2)$-appearance with high probability. A successful 3-leg as above will correspond to a $(1,2,3)$-appearance.

Working out the details, the success probability for one such sampled point in B, assuming that the right scale is known, is $\widetilde{\Omega}(\epsilon^3/m)$. Amplification can be done for all scales resulting in a high overall success probability and $\widetilde{O}(m/\epsilon^3)$ queries.

B.1.2 M_{12} is large

When M_{12} is large, Algorithm 2 will find a $(1,2,3)$-appearance with probability at least $1-n^{-\Omega(\log n)}$, by making $\widetilde{O}(m/\epsilon^2)$ queries.

Algorithm 2 $(1,2,3)$-freeness testing: M_{12} is large

Repeat $\widetilde{\Theta}(1/\epsilon)$ times:

1. Sample a uniformly random box B. Let R denote the row containing B and let R' be the restriction of the row to the boxes right of B.
2. For each scale $k \in [\log(n/m)]$,
   (a) Divide R equally into $n/(2^k m)$ horizontal strips.
   (b) Repeat $\widetilde{\Theta}(1)$ times:
      i. Run the $(1,2)$-freeness tester on B with proximity parameter $\epsilon'/(4\log n)$.
      ii. If the tester returns a $(1,2)$-appearance, let H be the horizontal strip containing the 2-leg of the $(1,2)$-appearance.
      iii. Let H' be the strip immediately above H (if it exists) and sample $\widetilde{\Theta}(m/\epsilon)$ points from $H' \cap R'$.
3. Return any $(1,2,3)$-appearance from among the points queried.

This is the hard case, and the proof of the correctness of Algorithm 2 can be found in Section 4.2.2.

B.2 M_2 is large

M_2 is the matching of $(1,2,3)$-appearances where the boxes containing the 1- and 2-legs do not share a row or column. Assume that M_2 has cardinality at least $\epsilon n^2/(3\log n)$. We partition $M_2 = M_{22} \cup M_{23}$ where in M_{22} the legs span 2 rows of boxes (the case of 2 columns is similar). M_{23} contains those $(1,2,3)$-appearances that span 3 rows and 3 columns.

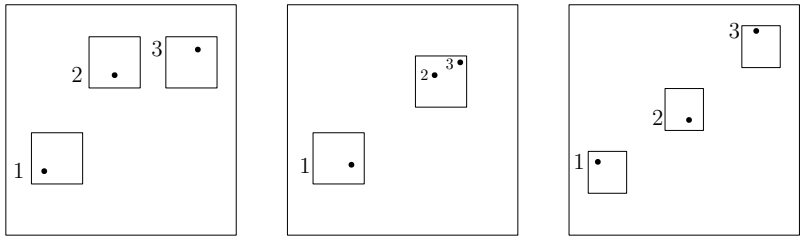
M_{22} is large. Assume first that $|M_{22}| \geq \epsilon n^2/(6\log n)$. There are only two cases as shown in (a) and (b) of Figure 7, and both cases are treated identically as follows. By our assumption, a uniformly random box B will contain, in expectation, $\Omega(\epsilon n^2/m^2)$ 2-legs from such $(1,2,3)$-appearances.

Therefore, with probability $\Omega(\epsilon)$, a uniformly random box B will contain $\Omega(\epsilon n^2/(m^2 \log n))$ many 2-legs from such $(1,2,3)$-appearances, which we call the 'good event' and condition on it. Let R' be the restriction of the row containing B to the boxes to the right of B including B. Conditioned on the good event, R' will have $\Omega(\epsilon n^2/(m^2 \log n))$ pairs that are the 2- and 3-legs of such $(1,2,3)$-appearances whose 2-leg is in B. In other words, R' is $\Omega(\epsilon/(m \log n))$-far from being $(2,3)$-free where the 2-legs are all in B. Hence, with probability at least $\Theta(1)$, our $(1,2)$-freeness tester with proximity parameter $\epsilon' = \Theta(\epsilon/(m \log n))$ will find such a $(2,3)$-appearance. Moreover, by Theorem 3.3, conditioned on finding a $(2,3)$-appearance from M_{22}, with probability $\widetilde{\Omega}(1)$, the 2-leg found will have value at least as large as the values of the 3/4 fraction of the 2-legs in B (i.e., the 2-leg found will have value in the "top quartile" of the values of all the 2-legs in B w.h.p.). Assuming this happens, let the legs of the $(2,3)$-appearance that is found be denoted by (p_2, p_3).

Let R'' denote the region of the grid formed by taking the union of all boxes to the left of B and below B. Now, there are $\Omega(\epsilon n^2/(m^2 \log n))$ points p_1 in the region R'' such that $p_1 \prec p_2$ and $f(p_1) < f(p_2)$. Hence, sampling $\widetilde{\Theta}(m^2/\epsilon)$ points in the region R'' will find such a point p_1 with probability $1 - 1/n^{\Omega(1)}$. Such a point p_1 together with (p_2, p_3) will form a $(1,2,3)$-appearance.

By a union bound over all bad events, we find a $(1,2,3)$-appearance with probability $\Omega(\epsilon)$. Thus, $\widetilde{\Theta}(1/\epsilon)$ iterations of the procedure outlined above guarantees the desired success probability.

The pseudocode for the procedure is given in Algorithm 3, and it has query complexity $\widetilde{\Theta}(m^2/\epsilon^2)$.



(a) M_{22} : Case (i) (b) M_{22} : Case (ii) (c) M_{23}

Figure 7: (a), (b) M_{22}: Boxes containing legs of $(1,2,3)$-appearances spanning 2 rows of $G_m^{(2)}$ (c) M_{23}: Boxes containing legs of $(1,2,3)$-appearances spanning 3 rows and 3 columns of $G_m^{(2)}$

Algorithm 3 $(1,2,3)$-freeness tester: M_{22} is large

Repeat $t = \widetilde{\Theta}(1/\epsilon)$ times:

1. Sample a box B uniformly at random from $G_m^{(2)}$.
2. Let R' denote the restriction of the row containing B to all the boxes to the right of B and including B. Let R'' denote the region of the grid formed by taking the union of all boxes to the left of B and below B. Namely, if B is to contain the 2-leg of an appearance from M_{32}, then the 3-leg is in R' and the 1-leg is in R''.
3. Run $t' = \widetilde{\Theta}(1/\epsilon)$ independent executions of a $(1,2)$-freeness tester on R' with proximity parameter $\epsilon' = \Omega(\epsilon/m)$.
4. Sample $t'' = \widetilde{\Theta}(m^2/\epsilon)$ points uniformly and independently at random from R''.
5. **Reject** if a $(1,2,3)$-appearance is found in the union of all points sampled in the steps above.

M_{23} is large. The case when $|M_{23}| = \Omega(\epsilon n^2/\log n)$ (see Figure 7(c)) is similar, except that the violation (p_2, p_3) should not be restricted to the row of the sampled box containing the 2-leg, but rather to the part of the grid consisting of boxes to the right and above the sampled box. This decreases the density of the matching to $\widetilde{\Omega}(\epsilon/m^2)$ but the overall complexity is the same as above.

B.3 M_3 is large

M_3 is the matching of $(1,2,3)$-appearances where the boxes containing the 1- and 2-legs share a column (resp. row), and the boxes containing the 2- and 3-legs share a row (resp. column). That is, the boxes containing the legs of each such $(1,2,3)$-appearance form a "Γ" (resp. "inverted Γ") shape. When M_3 is large, Algorithm 4 will find a $(1,2,3)$-appearance with probability at least $1 - n^{-\Omega(\log n)}$, by making $\widetilde{O}(m^2 \operatorname{poly}(1/\epsilon))$ queries. The detailed discussion of this case can be found in Section 4.2.4.

Algorithm 4 $(1,2,3)$-freeness tester: M_3 is large

For each $\ell \in \{0, 1, \ldots, \log n\}$:

1. Let $w = 2^{\ell-1}$ for $\ell \geq 1$ and $w = 1$ for $\ell = 0$.
2. Repeat $t = \widetilde{\Theta}(1/\epsilon)$ times:
   (a) Sample a box B uniformly at random from the grid $G_m^{(2)}$.
   (b) Let R' denote the restriction of $G_m^{(2)}$ to B and the boxes to the right of B and in the same row. Let C denote the restriction of $G_m^{(2)}$ to B and all the boxes below B and in the same column.
   (c) Run $\widetilde{\Theta}(m/\epsilon)$ independent iterations of the $(1,2)$-freeness tester with proximity parameter $\epsilon' = \Theta(\epsilon/(m \log n))$ on R'.
   (d) Consider a partition of C into subcolumns of width w each.
   (e) For each $(1,2)$-appearance (p_1, p_2) returned in Step (b)iii such that $p_1 \in B$, perform the following:
      i. Let p_1 belong to a subcolumn C' of C and let C'' be the subcolumn immediately to the left of C'.
      ii. If $\ell = 0$, then sample $\widetilde{\Theta}(m/\epsilon)$ indices uniformly and independently at random from the region of the subcolumn C' below B.
      iii. If $\ell \neq 0$, then sample $\widetilde{\Theta}(m/\epsilon)$ indices uniformly and independently at random from the region of the subcolumn C'' below B.
3. **Reject** if a $(1,2,3)$-appearance is found.

C Missing proofs from Section 6

In this section, we provide the missing proofs in Section 6, for the following lower bounds for nonadaptive and adaptive ϵ-testers for $(1,3,2)$-freeness on the hypergrid $[n]^2$.

Theorem C.1. *Any one-sided-error nonadaptive ϵ-tester for $(1,3,2)$-freeness on $[n]^2$ has query complexity $\Omega(n)$, for every $\epsilon \leq 1/81$.*

Theorem C.2. *Any one-sided-error adaptive ϵ-tester for $(1,3,2)$-freeness on $[n]^2$ has query complexity $\Omega(\sqrt{n})$, for every $\epsilon \leq 4/81$.*

Following [NRRS19], we reduce Theorem C.3 ("intersection-search") for two 2-*dimensional* $3n \times 3n$ 'monotone arrays' (w.r.t. $\prec$), to the problem of testing $(1,3,2)$-freeness on the 2-dimensional hypergrid $[9n]^2$.

Throughout this section, We denote the coordinates of an index $\mathbf{v}$ of a 2-dimensional array by (v_1, v_2). Vector addition and subtraction are coordinate-wise.

Problem C.3 (Intersection-Search for two 2-dimensional arrays)**.** *The input to this problem consists of two $3n \times 3n$ arrays A and B, and a constant γ with $0 \leq \gamma \leq 1$ (which we call the intersection parameter). A and B each consist of $9n^2$ distinct integers sorted in ascending order, with respect to the grid (partial) order $\prec$. It is promised that $|A \cap B| \geq \gamma \cdot (9n^2)$. The goal is to find $\mathbf{i}, \mathbf{j} \in [3n]^2$ such that $A[\mathbf{i}] = B[\mathbf{j}]$.*

The intersection-search problem can be reduced to the problem of testing $(1,3,2)$-freeness of real-valued functions on 2-dimensional hypergrids. We will now describe this reduction.

Lemma C.4. *Let (A, B, γ) be an input instance of Theorem C.3. Then, there exists a function $f\colon [9n]^2 \to \mathbb{R}$ that is at least $\gamma/9$-far from $(1,3,2)$-freeness, and there is a one-to-one correspondence between valid solutions of Theorem C.3, and $(1,3,2)$-appearances in f.*

Proof. Let (A, B, γ) be an input instance of Theorem C.3 – i.e., A and B are strictly increasing $3n \times 3n$ arrays, and have at least $\gamma \cdot (9n^2)$ elements in common. Let A^r be the reversal of A (i.e., $A^r[\mathbf{i}] := A[\mathbf{i}']$, where $\mathbf{i}' = (3n - i_1 + 1, 3n - i_2 + 1)$, if $\mathbf{i} = (i_1, i_2)$).

We now define a function $f\colon [9n]^2 \to \mathbb{R}$ in terms of the values in A and B as follows:

$$f(\mathbf{i}) = \begin{cases} A^r[\mathbf{i}] - \frac{1}{4}, & \text{if } \mathbf{i} = (2i_1 - 1, i_2),\ i_1, i_2 \in [3n], \\ A^r[\mathbf{i}] + \frac{1}{4}, & \text{if } \mathbf{i} = (2i_1, i_2),\ i_1, i_2 \in [3n], \\ B[\mathbf{i}], & \text{if } \mathbf{i} = (6n + i_1,\ 6n + i_2),\ i_1, i_2 \in [3n], \\ 0, & \text{otherwise.} \end{cases}$$

In the definition above, $\mathbf{i} = (i_1, i_2)$. Note that for every valid solution $(\mathbf{i}, \mathbf{j})$ of Theorem C.3 (i.e., for all indices $\mathbf{i}, \mathbf{j} \in [3n]^2$, such that $A[\mathbf{i}] = B[\mathbf{j}] = y$), the indices $(2i_1 - 1, i_2)$, $(2i_1, i_2)$, and $(6n + j_1, 6n + j_2)$, form a $(1,3,2)$-appearance in f, with f-values $y - 1/4$, $y + 1/4$, and y, respectively. It is also easy to check that these are the *only* kinds of $(1,3,2)$-appearances in f, as A^r is a strictly decreasing array consisting of positive integers, and B is a strictly increasing array of integers. Since every solution to Theorem C.3 uniquely defines one $(1,3,2)$-appearance in f and vice versa, f consists of at least $\gamma \cdot (9n^2)$ pairwise disjoint $(1,3,2)$-appearances, so that f is ϵ-far from $(1,3,2)$-freeness, where $\epsilon \geq (\gamma \cdot (9n^2))/(81n^2) = \gamma/9$. □

Now that we have the reduction Theorem C.4, we upper bound the success probability of any non-adaptive randomized algorithm for Theorem C.3, which would then directly give us Theorem C.1.

Note that we only consider algorithms which output a pair of indices $\mathbf{i}, \mathbf{j} \in [3n]^2$ only if it actually queries $A[\mathbf{i}]$ and $B[\mathbf{j}]$, and $A[\mathbf{i}] = B[\mathbf{j}]$. It outputs "fail" if it does not actually hit such indices.

Lemma C.5. *Let $\mathcal{A}$ be any nonadaptive randomized algorithm for the intersection-search problem Theorem C.3, with intersection parameter $\gamma \geq 1/9$. If $\mathcal{A}$ makes q queries and $q < n^2/2$, then the success probability of $\mathcal{A}$ is at most $q^2/(4n^2)$.*

Proof. We use Yao's minimax principle [Yao77] to prove the lemma. We define a distribution $\mathcal{D}$ on the valid inputs to Theorem C.3, and then show that any *nonadaptive deterministic* algorithm for this problem will succeed (i.e., output a pair $(\mathbf{i}, \mathbf{j})$ such that $A[\mathbf{i}] = B[\mathbf{j}]$) with probability at most $q^2/(4n^2)$ – i.e., the proportion of inputs (under the distribution $\mathcal{D}$) for which the algorithm outputs a pair of indices is at most $q^2/(4n^2)$.

We now describe the distribution $\mathcal{D}$ from which we sample an input instance to Theorem C.3, with $\gamma = 1/9$ – i.e., two monotone increasing (with respect to $\prec$) $3n \times 3n$ integer arrays A and B, with exactly n^2 common elements. To this end, (i) pick a $3n \times 3n$ array X with 0-1 entries uniformly at random, (ii) independently pick a set $S \subset [2n]^2$ of size n^2, uniformly at random, and (iii) independently pick an index $\mathbf{k} = (k_1, k_2) \in [n]^2$ uniformly at random.

The idea behind these random choices is to construct the input arrays A and B such that $A[\mathbf{i}] = B[\mathbf{j}]$ if and only if $\mathbf{i} \in S$, and $\mathbf{j} = \mathbf{i}+\mathbf{k}$, where '+' refers to coordinate-wise addition. Since S is picked uniformly at random, and $q < n^2/2$, it will not be possible for any nonadaptive algorithm to determine S completely. Furthermore, we'll use the randomly chosen 0-1 entries in the array X, to make sure that any nonadaptive deterministic algorithm outputs a pair $(\mathbf{i}, \mathbf{j})$ with $A[\mathbf{i}] = B[\mathbf{j}]$, only if it actually hits the indices $\mathbf{i}$ and $\mathbf{j}$. Choosing $\mathbf{k}$ randomly ensures that such hits are unlikely, if $q = o(n)$.

Given these random choices, the two input arrays A and B are defined as follows:

$$A[\mathbf{i}] := 6n(i_1 + i_2) + 2i_1 + X[\mathbf{i}], \quad \text{for } \mathbf{i} = (i_1, i_2) \in [3n]^2.$$

and

$$B[\mathbf{i}] := \begin{cases} 6n((i_1 - k_1) + (i_2 - k_2)) + 2(i_1 - k_1) + X[\mathbf{i}-\mathbf{k}], & \text{if } \mathbf{i}-\mathbf{k} \in S, \\ 6n((i_1 - k_1) + (i_2 - k_2)) + 2(i_1 - k_1) + (1 - X[\mathbf{i}-\mathbf{k}]), & \text{if } \mathbf{i} - \mathbf{k} \in [3n]^2 \setminus S, \text{ and} \\ 6n((i_1 - k_1) + (i_2 - k_2)) + 2(i_1 - k_1), & \text{otherwise} \end{cases}$$

It can be verified that the arrays A and B consist of $9n^2$ distinct integers each, and are sorted in ascending order with respect to the grid (partial) order $\prec$ on the indices. Further, A only consists of positive integers. Note that $A[\mathbf{i}] = B[\mathbf{j}]$ for some indices $\mathbf{i}, \mathbf{j} \in [3n]^2$, if and only if $\mathbf{i} \in S$, and $\mathbf{j} = \mathbf{i}+\mathbf{k}$. Hence, A and B have exactly n^2 common values. Also, given A and B, for every $\mathbf{i} \in [2n]^2$, any nonadaptive algorithm can find such a common value only if it queries these indices, when the input is drawn from this distribution.

Let $\mathcal{A}$ be a nonadaptive deterministic algorithm for Theorem C.3, with input (A, B) drawn from the given distribution $\mathcal{D}$, and $\gamma = 1/9$. Let $\mathcal{A}$ query the array A at the set of indices $Q_A \subset [3n]^2$, and query B at the set of indices $Q_B \subset [3n]^2$. Let $q = |Q_A| + |Q_B|$. Let $D = \{\mathbf{j}-\mathbf{i} : \mathbf{i} \in Q_A, \mathbf{j} \in Q_B\}$. Then, if $\mathcal{A}$ does indeed output a pair of indices $(\mathbf{i}, \mathbf{j})$ (i.e., it succeeds), then it must be the case that $\mathbf{k} \in D$. Hence, the success probability of $\mathcal{A}$ is at most $\Pr[\mathbf{k} \in D]$, which is at most $q^2/(4n^2)$, as $|Q_A| \cdot |Q_B| \leq q^2/4$, and $\mathbf{k}$ has been picked uniformly at random from $[n]^2$. This completes the proof. □

Hence, any nonadaptive randomized algorithm must query $\Omega(n)$ indices from A and B so as to solve Theorem C.3 with probability $\Theta(1)$, provided $\gamma = \Theta(1)$. The $\Omega(n)$-query lower bound Theorem C.1 directly follows from Theorem C.4 and Theorem C.5. □

The adaptive lower bound Theorem C.2 is directly obtained by first proving an $\Omega(\sqrt{n})$-query lower bound for *adaptive* randomized algorithms for Theorem C.3, that succeed with probability $\Theta(1)$.

Lemma C.6. *Let $\mathcal{A}$ be any adaptive randomized algorithm for the intersection-search problem Theorem C.3, with intersection parameter $\gamma \geq 4/9$. If $\mathcal{A}$ makes q queries, then the success probability of $\mathcal{A}$ is at most $q^2/(4n+4)$.*

Proof. As was done in Theorem C.5, we use Yao's minimax principle to prove the lemma. We define a distribution $\mathcal{D}$ on the valid inputs to Theorem C.3, and then show that any *adaptive deterministic* algorithm for this problem will succeed with probability at most $q^2/(4n+4)$ – i.e., the proportion of inputs (under the distribution $\mathcal{D}$) for which the algorithm outputs a pair of indices is at most $q^2/(4n+4)$. We first describe the intuition behind the construction of such a distribution.

Let $(A, B, \gamma \geq 4/9)$ be an input instance of Theorem C.3. Note that the set of indices $[3n]^2$ may be partitioned into $6n-1$ antichains $\{S_r\}_{r=2}^{6n}$, where $\mathrm{S}_r := \left\{\mathbf{x}, \mathbf{y} \in [3n]^2 : \; x_1 + x_2 = y_1 + y_2 = r\right\}$. Note that S_{3n+1} is the main diagonal.

The idea is to construct the distribution in such a way that the corresponding antichains S_r in A and B contain the same set of elements for each r, and for different r's, the antichains are disjoint. This ensures that all solutions to the intersection-search problem belong to the same antichain. We will focus only on $r \in \{n+2, \ldots, 3n+1\}$, for which $|S_r| = r - 1 = \Theta(n)$. Hence, the union of all these antichains constitute $4n^2 + n$ elements, implying that $\gamma > 4/9$.

To ensure that the matrices A and B are monotone with respect to $\prec$, we will pick the elements in S_r (in both A and B, as explained), much larger than the ones in S_{r-1}. Further, we will make sure that the elements outside our antichains of interest cannot be matched. Now, since each S_r is an antichain, any order on its elements is consistent with the grid order $\prec$. Therefore, to find a match, one really needs to solve birthday-search (Theorem 6.4) on any one of the S_r ($r \in \{n+2, \ldots, 3n+1\}$).

The input distribution $\mathcal{D}$ may be formally defined as follows – for each $r \in \{n+2, \ldots, 3n+1\}$, independently sample a permutation $\pi_r \in \mathcal{S}_{r-1}$ uniformly at random (i.e., a random permutation on $r-1$ letters). Then, the input arrays A and B are defined as follows:

$$A[\mathbf{i}] := 6n(i_1 + i_2) + 2i_1, \quad \text{for } \mathbf{i} = (i_1, i_2) \in [3n]^2.$$

and

$$B[\mathbf{i}] := \begin{cases} 6n(i_1 + i_2) + 2\pi_r(i_1), & \text{if } r = i_1 + i_2 \in \{n+2, \ldots, 3n+1\}, \\ 6n(i_1 + i_2) + 2i_1 + 1, & \text{otherwise} \end{cases}$$

It can be verified that A and B consist of $9n^2$ distinct elements each, are sorted in ascending order. A and B contain at least $4n^2$ common elements, and for each such element $y = A[\mathbf{i}] = B[\mathbf{j}]$, $\mathbf{i}$ and $\mathbf{j}$ belong to the same antichain $S_{i_1+i_2}$ ($= S_{j_1+j_2}$), where $n+2 \leq i_1 + i_2 = j_1 + j_2 \leq 3n+1$.

Let $\mathcal{A}$ be any adaptive deterministic algorithm for Theorem C.3. Consider an antichain S_r, where $r \in \{n+2, \ldots, 6n\}$. Let $\mathcal{A}$ query A at the indices $Q_A^{(r)} \subset S_r$, and query B at the indices $Q_B^{(r)} \subset S_r$. Let $q_r = \left|Q_A^{(r)}\right| + \left|Q_B^{(r)}\right|$. Then, the probability (under the given input distribution $\mathcal{D}$) that $\mathcal{A}$ finds a valid pair of indices on S_r, with $r \in \{n+2, \ldots, 3n+1\}$, is at most $\frac{\left|Q_A^{(r)}\right| \cdot \left|Q_B^{(r)}\right|}{|S_r|} \leq \frac{q_r^2}{4n+4}$, as $\left|Q_A^{(r)}\right| \cdot \left|Q_B^{(r)}\right| \leq q_r^2/4$, and $|S_r| \geq n+1$. The success probability is 0 if $r \notin \{n+2, \ldots, 3n+1\}$. A union bound over these good events over all the antichains (i.e., $r \in \{n+2, \ldots, 6n\}$) finishes the proof. ∎

Hence, any adaptive randomized algorithm must query $\Omega(\sqrt{n})$ indices from A and B so as to solve Theorem C.3 with probability $\Theta(1)$, provided $\gamma = \Theta(1)$. The $\Omega(\sqrt{n})$-query lower bound Theorem C.2 directly follows from Theorem C.4 and Theorem C.6. □

D Layering and Gridding

In this section, we describe an algorithm that we call Gridding, which is a common subroutine in all of our algorithms. This machinery is from [NV24], tailored to the 2-dimensional domain $[n]^2$. The output of Gridding, given oracle access to the function $f\colon [n]^2 \to \mathbb{R}$ and a parameter $m \le n$, is an $m \times m \times m$ grid of boxes that partitions either the grid $[n]^2 \times R(f)$ or a region inside of it into boxes, with the property that the density of each box, which we define below, is "well controlled".

Definition D.1 (Density of a box)**.** *Consider index and value subsets $S = X \times Y \subseteq [n]^2$ and $I \subseteq R(f)$, respectively, where $X \subseteq [n]$ and $Y \subseteq [n]$. The density of $\mathsf{box}(S, I)$, denoted by $\mathsf{den}(S, I)$, is the number of points in $\mathsf{box}(S, I)$ normalized by its size $|S|$.*

Definition D.2 (Nice partition of a box)**.** *For index and value sets $S = X \times Y \subseteq [n]^2$ and $I \subseteq R(f)$ and a parameter $m \le n$, we say that $\mathcal{I} = \{I_1, I_2, \ldots, I_{m'}\}$ forms a nice m-partition of $\mathsf{box}(S, I)$ if*

- *$m' \le 2m$,*
- *$I_1, \ldots, I_{m'}$ are pairwise disjoint, and $\bigcup_{j \in [m']} I_j = I$. In particular, the largest value in I_j is less than the smallest value in $I_{j'}$ for $j < j'$.*
- *for $j \in [m']$, either $\mathsf{den}(S, I_j) < 4/m$ OR I_j contains exactly one value and $\mathsf{den}(S, I_j) \ge 1/2m$. In the first case, we say that $\mathsf{box}(S, I_j)$ is a* multi-valued layer *of $\mathsf{box}(S, I)$, and in the second case, we say that $\mathsf{box}(S, I_j)$ is a* single-valued layer *of $\mathsf{box}(S, I)$.*

D.1 Layering

The main part of Gridding is an algorithm Layering which is described in Algorithm 5. A similar algorithm was used by Newman and Varma [NV21] for estimating the length of the longest increasing subsequence in an array. $\mathsf{Layering}(S, I, m)$ is given $S = X \times Y \subseteq [n]^2$, $I \subseteq R(f)$, and $m \le n$ as inputs, and it outputs, with probability at least $1 - 1/n^{\Omega(\log n)}$, a set $\mathcal{I}$ of intervals that is a nice m-partition of $\mathsf{box}(S, I)$. It works by sampling $\widetilde{O}(m^2)$ points from $\mathsf{box}(S, I)$ and outputs the set $\mathcal{I}$ based on these samples. Note that the sets X, Y, and I are either contiguous index/value intervals. Additionally, we always apply the algorithm Layering to boxes of density $\Omega(1/\log n)$.

Claim D.3. *If $\mathsf{den}(S, I) > 1/\log n$, then with probability $1 - 1/n^{\Omega(\log n)}$, Layering(S, I, m) returns a collection of intervals $\mathcal{I} = \{I_j\}_{j=1}^{m'}$ such that $\mathcal{I}$ is a nice m-partition of $\mathsf{box}(S, I)$. Furthermore, it makes a total of $m \log^4 n$ queries.*

Proof. Since $\mathsf{den}(S, I) > 1/\log n$, at least $m \log^2 n$ of the sampled points fall in $\mathsf{box}(S, I)$, with probability at least $1 - \exp(-(m \log^3 n)/8)$, by a Chernoff bound. Hence, the algorithm does not fail in Algorithm 5. In the rest of the analysis, we condition on this event happening.

The total number of intervals of weight at least u/m is at most m since the total weight is u. Other intervals have weight less than u/m and for each such interval I_j, it must be the case that I_{j-1} and I_{j+1} are of weight at least u/m. It follows that $m' \le 2m$.

Algorithm 5 Layering(S, I, m)

1: Sample a set of $m \log^4 n$ indices from S uniformly and independently at random.

2: Let U denote the set of points in the sample that belong to $\mathsf{box}(S, I)$. If $u := |U| < m \log^2 n$, then **FAIL**.

3: We sort the multiset of values $V = \{f(p) : \ p \in U\}$ to form a strictly increasing sequence $\mathsf{seq} = (v'_1 < \ldots < v'_q)$, where, for each $i \in [q]$, we associate a weight w_i that equals the multiplicity of v'_i in the multiset V of values. ▷ Note that $\sum_{i \in [q]} w_i = u$.

4: We now partition the sequence $W = (w_1, \ldots, w_q)$ into maximal disjoint contiguous subsequences $W_1, \ldots W_{m'}$ such that for each $j \in [m']$, either $\sum_{w \in W_j} w < 2u/m$, or W_j contains only one member w for which $w > u/m$. ▷ This can be done greedily as follows. If $w_1 > u/m$, then W_1 will contain only w_1, and otherwise, W_1 will contain the maximal subsequence $(w_1, \ldots, w_i)$ whose sum is at most $2u/m$. We then delete the members of W_1 from W and repeat the process. For $i \in [m']$, let $w(W_i)$ denote the total weight in W_i.

Correspondingly, we obtain a partition of the sequence seq of sampled values into at most m' subsequences $\{\mathsf{seq}_j\}_{j \in [m']}$. Some subsequences contain only one value of weight at least u/m and are called *single-valued*. The remaining subsequences are called *multi-valued*.

For a subsequence seq_j, let $\alpha_j = \min(\mathsf{seq}_j)$ and $\beta_j = \max(\mathsf{seq}_j)$. Let $\beta_0 = \inf(I)$. Note that $\alpha_j \leq \beta_j$ and $\beta_{j-1} < \alpha_j$ for all $j \in [m']$.

5: For $j \in [m']$, we associate with the subsequence seq_j, an interval $I_j \subseteq \mathbb{R}$, where $I_j = (\beta_{j-1}, \beta_j] \cap I$, and an approximate density $\widetilde{\mathsf{den}}(S, I_j) = w(W_j)/u$. The interval is multi-valued or single-valued depending on whether its corresponding sequence is multi-valued or single-valued, respectively.

6: **Return** the set $\mathcal{I} = \bigcup_{\ell \in [m']} I_\ell$ of the m' intervals.

We now prove that the family $\mathcal{I}$ output by Layering is a nice m-partition of $\mathsf{box}(S, I)$. It is clear from the description of Algorithm 5 that the intervals output by the algorithm are disjoint. Let $\mathcal{B} = \{[a, b] : a, b \in I \text{ and } \exists v, w \in S \text{ such that } f(v) = a, f(w) = b\}$ denote the set of all true intervals of points from $\mathsf{box}(S, I)$. Consider an interval $[a, b] \in \mathcal{B}$ such that $\mathsf{den}(S, [a, b]) \geq 4/m$. The probability that less than $2u/m$ points from the sample have values in the range $[a, b]$ is at most $1/n^{\Omega(\log n)}$ by a Chernoff bound. Conditioning on this event *not* occurring implies that for every $I_j, j \in [m']$ output as a multi-valued interval by the algorithm, we have $\mathsf{den}(S, I_j) < 4/m$. Finally, for a single-valued interval $[a, a] \in \mathcal{B}$ such that $\mathsf{den}(S, [a, a]) < 1/2m$, with probability at least $1 - 1/n^{\Omega(\log n)}$, we have $\widetilde{\mathsf{den}}(S, [a, a]) \leq \frac{3}{2}\mathsf{den}(S, [a, a]) < 3/4m$, where $\widetilde{\mathsf{den}}(S, I')$ denotes the estimated density (as estimated in Algorithm 5) for a layer $\mathsf{box}(S, I')$ when $I' \subseteq I$. Conditioning on this event occurring implies that for every $I_j, j \in [m']$ output as a single-valued interval by the algorithm, we have $\mathsf{den}(S, I_j) \geq 1/2m$.

The claim about the query complexity is clear from the description of the algorithm. □

D.2 Gridding

Next, we describe the algorithm Gridding.

Algorithm 6 Gridding(S, I, m, β)

Input: $X \subseteq [n]$ is a union of disjoint x-stripes, $Y \subseteq [n]$ is a union of disjoint y-stripes, so that $S := X \times Y$ is a disjoint union of rectangles in the xy-plane, $I \subseteq R(f)$ is a disjoint union of intervals of values in $R(f)$, m is a parameter defining the 'coarse' grid size, $\beta < 1$ is a density threshold.

Output: A grid of boxes $G^{(3)}_{m'}$, $m' \leq 2m$ in which there will be $\widetilde{O}(m^2)$ marked boxes.

1: Call Layering (Algorithm 5) on inputs S, I, m. This returns, with high probability, a set $\mathcal{I}$ of $m' \leq 2m$ value intervals $I = \bigcup_{j \in [m']} I_j$ that forms a nice m-partition of $\mathsf{box}(S, I)$.

2: Partition X into m' contiguous intervals $X_1, \ldots X_{m'}$ each of size $|X|/m'$ and partition Y into m' contiguous intervals $Y_1, \ldots Y_{m'}$ each of size $|Y|/m'$. This defines the grid of boxes $G^{(3)}_{m'} = \{\mathsf{box}(X_i \times Y_j, I_k) : (i, j, k) \in [m']^3\}$ inside the larger box $\mathsf{box}(S, I)$. For each $(i, j) \in [m']^2$, denote by $S_{i,j}$ the rectangle $X_i \times Y_j$ in the xy-plane.

3: Sample and query, independently at random, $\frac{\log^4 n}{\beta^2}$ points from each rectangle $S_{i,j}, (i, j) \in [m']^2$. For each $(i, j, k) \in [m']^3$, if $\mathsf{box}(S_{i,j}, I_k)$ contains a sampled point, then tag that box as *marked*. If $\mathsf{box}(S_{i,j}, I_k)$ contains at least $3\beta/4$ fraction of the sampled points in the rectangle $S_{i,j}$, tag that box as *dense*.

4: **Return** the grid $G^{(3)}_{m'}$ along with the tags on the various boxes.

We note that initially, at the topmost recursion level of the algorithm for π-freeness, we call Gridding with $S = [n]^2$, $I = (-\infty, +\infty)$ and our preferred m which is typically $m := n^\delta$, for some small $\delta < 1$.

We prove in Theorem D.4 that running Gridding(S, I, m) results in a partition of $\mathsf{box}(S, I)$ into a grid of boxes $G^{(3)}_{m'}$ in which either the marked boxes contain a π-appearance, or the union of points in the marked boxes contain *all* but an η fraction of the points in $[n]^2 \times R(f)$, for $\eta \ll \epsilon$. Additionally, with high probability, all boxes that are tagged *dense* have density at least $\frac{1}{8}$-th of the threshold β for marking a box as dense.

Claim D.4. $\mathsf{Gridding}(S, I, m, \beta)$ *returns a grid of boxes* $G^{(3)}_{m'}$ *that decomposes* $\mathsf{box}(S, I)$*. It makes* $\widetilde{O}(m^2/\beta^2)$ *queries, and with high probability,*

- *The set of intervals corresponding to the layers of* $G^{(3)}_{m'}$ *form a nice* m*-partition of* I*.*
- *For every* $(i, j) \in [m']^2$*, either* $\mathsf{box}(S_{i,j}, I)$ *contains at least* $\frac{\log^2 n}{100\beta^2}$ *marked boxes, or the number of points in the marked boxes in* $\mathsf{box}(S_{i,j}, I)$ *is at least* $(1 - \frac{1}{\log^2 n}) \cdot |S_{i,j}|$*.*
- *Every box that is tagged dense has density at least* $\beta/8$*, and every box of density at least* β *is tagged as dense.*

Proof. The bound on the query complexity as well as the first item follows directly from Theorem D.3. The third item follows by a simple application of the Chernoff bound followed by a union bound over all rectangles.

For the second item, fix a rectangle $S_{i,j}$ of $G^{(3)}_{m'}$. Let $T \subseteq [m']$ be the set of all $k \in [m']$ such that $\mathsf{box}(S_{i,j}, I_k)$ gets marked during Step 3 in $\mathsf{Gridding}$. If $\sum_{k \in T} \mathsf{den}(S_{i,j}, I_k) \geq 1 - 1/(\log^2 n)$, we are done. Otherwise, each query independently hits a box that is not marked by any of the previous queries with probability greater than $1/(\log^2 n)$. Thus, the expected number of boxes marked is at least $\frac{\log^2 n}{\beta^2}$. Chernoff bound implies that, with probability at least $1 - n^{-\Omega(\log n)}$, at least $\frac{\log^2 n}{100\beta^2}$ boxes are marked. The union bound over all the rectangles implies the second item. □

E Deletion and Hamming distances

In this section, we prove relationships between deletion and Hamming distances from pattern freeness for various kinds of order patterns that we have discussed in the paper. In Appendix E.1, we prove (Theorem 2.2) that these distances are equal for certain types of permutation patterns over high-dimensional hypergrids. We also describe an example (Theorem 2.4) involving a 4-pattern for which deletion and Hamming distances are significantly different even for the case of 2-dimensional grids.

In the case of patterns which are *not* permutations, these distances need not be equal even for order patterns of length 3. In Appendix E.2, we give an example of a "1, 2, 3-fork" pattern for which the gap between the Hamming and deletion distances is large.

E.1 Permutation patterns

First, we prove (Theorem 2.2) that deletion and Hamming distances are equal for all permutation patterns π (of any length) for which $\{\pi(1), \pi(k)\} \cap \{1, k\} \neq \emptyset$, and for all $f\colon [n]^d \to \mathbb{R}$. We note that this case subsumes all monotone patterns (of any length), and all patterns of length at most 3 – this is clear either by looking at the pattern upside down or by multiplying all f-values by -1.

Claim E.1. (Restatement of Theorem 2.2) *Let* $k \geq 1$ *and* $\pi \in \mathcal{S}_k$ *with* $\{\pi(1), \pi(k)\} \cap \{1, k\} \neq \emptyset$*. Let* $n, d \geq 1$*. The Hamming distance of* $f\colon [n]^d \to \mathbb{R}$ *to* P^d_π *is equal to the deletion distance of* f *to* P^d_π*.*

Proof. Let $f\colon [n]^d \to \mathbb{R}$, and $\pi \in \mathcal{S}_k$ with $\pi(1) = 1$. The other cases are analogous, either by looking at the domain upside down, or by multiplying the f-values by -1. It is easy to see that the deletion

distance of f to P_π^d is at most the Hamming distance – if changing the f-values on some $S \subseteq [n]^d$ makes it π-free, then $f\big|_{[n]^d \setminus S}$ is π-free. We will prove the reverse inequality.

Let Δ be the deletion distance of f to P_π^d, and let $S := \{i_1, \ldots, i_\Delta\}$ be a set of indices in $[n]^d$ such that $f\big|_{[n]^d \setminus S}$ is π-free.

Define a function $f^*\colon [n]^d \to \mathbb{R}$ such that $f^*\big|_{[n]^d \setminus S} := f\big|_{[n]^d \setminus S}$ and for all $j \in [\Delta]$,

$$f^*(i_j) := \min\{f(p)\colon p \notin S \text{ and } p \prec i_j\}$$

We will now show that f^* is π-free. This will imply that the Hamming distance of f to P_π^d is at most Δ, which will complete the proof. Note that we may assume that $\bar{1} = (1, \ldots, 1) \notin S$, for if $\bar{1} \in S$, setting $f^*(\bar{1}) := \max\{f(p)\colon p \notin S\}$ would ensure that $\bar{1}$ cannot be a leg of any π-appearance in f^*.

To show f^* is π-free, we let $j \in \{0, \ldots, \Delta\}$, and prove that $f^*\big|_{[n]^d \setminus \{i_{j+1}, \ldots, i_\Delta\}}$ is π-free. For $j = \Delta$, we interpret this to mean f^* is π-free, which is what we set out to prove. The proof proceeds by induction on j. The base case $j = 0$ follows directly from the definition of f^* as $f^*\big|_{[n]^d \setminus S} = f\big|_{[n]^d \setminus S}$, and $f\big|_{[n]^d \setminus S}$ is π-free.

Now, let us assume that $j \geq 1$ and $f^*\big|_{[n]^d \setminus \{i_j, \ldots, i_\Delta\}}$ is π-free but that $f^*\big|_{[n]^d \setminus \{i_{j+1}, \ldots, i_\Delta\}}$ is not π-free. Suppose i_j was the 1-leg of a π-appearance $(i_j, p_2, \ldots, p_k)$ in $f^*\big|_{[n]^d \setminus \{i_{j+1}, \ldots, i_\Delta\}}$, where $p_2, \ldots, p_k \in [n]^d \setminus \{i_{j+1}, \ldots, i_\Delta\}$ and $i_j \prec p_2 \prec \ldots \prec p_k$. From the way we defined $f^*(i_j)$, there exists $p \prec i_j$ such that $f^*(p) = f^*(i_j)$, so that $(p, p_2, \ldots, p_k)$ is a π-appearance in $f^*\big|_{[n]^d \setminus \{i_j, \ldots, i_\Delta\}}$, contradicting the induction hypothesis. Furthermore, i_j cannot be the t-leg (for any $2 \leq t \leq k$) of any π-appearance in $[n]^d \setminus \{i_{j+1}, \ldots, i_\Delta\}$ since, for all $p \prec i_j$, $f^*(i_j) \leq f(p)$. This completes the proof. □

We will now give an example of a pattern $\pi \in \mathcal{S}_4$ and a function $f\colon [n]^2 \to \mathbb{R}$ for which the difference between the Hamming and deletion distances of f from π-freeness is $\Theta(n)$. This example generalizes straightforwardly to longer patterns and higher dimensions.

Claim E.2. (Restatement of Theorem 2.4) *There exists a 4-pattern $\pi \in \mathcal{S}_4$ and a function $f\colon [n]^2 \to \mathbb{R}$ such that the deletion distance of f from π-freeness is 1 and the Hamming distance of f from π-freeness is $\Theta(n)$.*

Proof of Theorem 2.4. Let $\pi := (2, 4, 1, 3) \in \mathcal{S}_4$. Consider the function $f\colon [n]^2 \to \mathbb{R}$ whose f-values are represented by the matrix

$$\begin{bmatrix}
 & & & 6 & & & & & & \\
 & 8 & & \vdots & & & 8 & & & \\
 & & & 6 & & & & & & \\
2 & \cdots & 2 & \mathbf{4} & 1 & \cdots & 1 & 3 & \cdots & 3 \\
 & & & 7 & & & & & & \\
 & & & \vdots & & & & & & \\
 & 8 & & 7 & & & 8 & & & \\
 & & & 5 & & & & & & \\
 & & & \vdots & & & & & & \\
 & & & 5 & & & & & &
\end{bmatrix}$$

Let the number of occurrences of 1, 2, and 3 on the "special" row in the matrix each be $\Theta(n)$. Similarly, let the number of occurrences of 5, 6, and 7 on the "special" column in the matrix each be $\Theta(n)$. There are thus $\Theta(n)$ π-appearances on the "$2, \mathbf{4}, 1, 3$"-row. Similarly, on the "$5, 7, \mathbf{4}, 6$" column, there are $\Theta(n)$ π-appearances.

The deletion distance of f from being π-free is 1 as the deletion of the **4** marked in bold in the matrix is sufficient to remove all the π-appearances. However, the Hamming distance of f from being π-free is $\Theta(n)$ since the modification of the **4** cannot simultaneously remove the π-appearances on the row and the column to which it belongs. This is because the **4** in the matrix is simultaneously the 4-leg of each of the π-appearances along the row, and the 1-leg of each of the π-appearances along the column. Hence, we must modify $\Theta(n)$ entries either on the row or the column to remove all the π-appearances. □

E.2 Non-permutation patterns

As mentioned earlier, when $d = 1$, every pattern is a permutation pattern. However, when $d \geq 2$, order patterns that are not permutations exist, since the partial order $\prec$ with which we work, is not a total order on the domain $[n]^d$ of the function being tested. As already remarked, even for $d = 2$, there are 3-patterns (which are not permutations) for which the Hamming distance may be arbitrarily larger than the deletion distance. We will give an example of such a pattern which we call the "$1, 2, 3$-fork" pattern.

A function $f\colon [n]^d \to \mathbb{R}$ is said to contain a $1, 2, 3$-fork pattern if there exist $p_1, p_2, p_3 \in [n]^d$ such that $p_2 \prec p_1$, $p_2 \prec p_3$, p_1 and p_3 are not comparable with respect to $\prec$, and $f(p_1) < f(p_2) < f(p_3)$. As usual, we call p_1, p_2, and p_3 the 1-leg, 2-leg, and the 3-leg of the $1, 2, 3$-fork pattern that they form, respectively. For the rest of this section, let π denote the $1, 2, 3$-fork pattern.

Claim E.3. *There exists a function $f\colon [n]^2 \to \mathbb{R}$ such that the deletion distance of f from π-freeness is 1 and the Hamming distance of f from π-freeness is $\Theta(n)$.*

Proof. Consider the function $f\colon [n]^2 \to \mathbb{R}$ whose f-values are represented by the matrix

$$\begin{bmatrix} 1 & \cdots & 1 & 4 & 4 \\ 2 & \cdots & 2 & \mathbf{3} & 2 \\ & & & 4 & 5 \\ & 5 & & \vdots & \vdots \\ & & & 4 & 5 \end{bmatrix}$$

Note that there are $\Theta(n)$ π-appearances formed by the values 1, 2, and **3**, where the **3** marked in bold is the 3-leg of each of these π-appearances. There are also $\Theta(n)$ π-appearances formed by the values **3**, 4, and 5, where the **3** marked in bold is the 1-leg of each of these π-appearances. The deletion distance of f from π-freeness is 1 as deleting the **3** would suffice. However, the Hamming distance of f from π-freeness is $\Theta(n)$, thanks to **3** simultaneously being the 1-leg and the 3-leg of $\Theta(n)$ π-appearances each. □

We do not know how to design tests for such patterns with respect to the Hamming distance and conjecture that it requires $n^{\omega(1)}$ queries. This is to be compared with the only known provable lower bound on permutation-freeness testing which is $\Omega(\log n)$.

Thus, it makes sense to test freeness of such order patterns with respect to the deletion distance. Our methods for $(1,3,2)$-freeness apply for such patterns too, and in particular, an $\widetilde{O}(n)$-query erasure-resilient test can be constructed along the lines of Section 5.2.