TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Raymond and Beverly Sackler Faculty of Exact Sciences
Blavatnik School of Computer Science

Fast Algorithms in Highly Structured Settings

Thesis submitted in partial fulfillment of the requirements for the degree "Doctor of Philosophy"

by

Omri Ben-Eliezer

Under the supervision of
Prof. Noga Alon

June 2020

Abstract

The field of property testing studies what can be deduced from data given limited access to it. While research in property testing has blossomed in the last two and a half decades, a large majority of the work until now has been devoted to data that is relatively unstructured, like probability distributions, or to properties with an inherent underlying symmetry, like graph properties (in which the labels are assumed to "not matter", a symmetry assumption that might not always be realistic). In contrast, the understanding of property testing in highly structured settings, where symmetry cannot be assumed and utilized, is more limited.

This thesis focuses on property testing in the highly structured regime. It develops several new tools and frameworks which advance the state of the art in several central fronts of structured property testing, and exhibits surprising combinatorial phenomena that arise in the study of problems in this regime. The contributions are divided into three main lines of work.

The first part concerns property testing in ordered graphs. A well-known result by Alon and Shapira asserts that any hereditary unordered graph property (where the labels of the vertices do not matter) is testable with a constant number of queries. The proof of this statement, however, heavily relies on the underlying symmetry of the problem, and does not apply to graphs where the labels are structured (are ordered, say, or represent the rows and columns of an image, where pixel locations are important). It was asked by Alon, Fischer and Newman in 2007 whether these results can be extended to the structured setting. We answer this affirmatively, developing Szemerédi-regularity schemes that are suitable for ordered settings.

The second part considers detection of global structural patterns in sequential data. Given a sequence of real numbers that contains many disjoint copies of a (constant-length) structural pattern, how can one detect one copy of the pattern efficiently? As it turns out, this problem lends itself to beautiful combinatorial structure, whose investigation sheds light on its algorithmic understanding. This answers a recent open question by Newman, Rabinovich, Rajendraprasad, and Sohler, and yields tight adaptive and non-adaptive testing algorithms for monotone patterns (curiously, the non-adaptive query complexity for monotone patterns of fixed length k in data of length n is $\Theta((\log n)^{\lfloor \log_2 k \rfloor})$), as well as lower bounds showing that for almost all patterns, the effectiveness of non-adaptive algorithms beyond naive uniform sampling is negligible.

The third part studies local properties in structured multi-dimensional data from the property testing perspective; roughly speaking, a property is local if it is characterized by small forbidden consecutive substructures, a definition that captures many previously investigated properties in the structured regime. The main result is that any local property of multi-dimensional arrays is testable with a sublinear number of queries via a canonical non-adaptive testing algorithm, querying sphere-like structures in the data. This generic approach is widely optimal for one-dimensional arrays, even among adaptive methods, and for some high-dimensional properties it is optimal among non-adaptive algorithms. It provides the first known sublinear-query test for challenging properties like convexity or submodularity in high dimensions. In addition, we prove a combinatorial *modification lemma* on the structure of local properties, which allows us to prove the efficient testability of pattern matching type properties. This answers a question of Fischer and Newman from 2001.

Acknowledgments

The journey towards this thesis has been a delightful and illuminating experience. The best part has been the opportunity to meet many unique, intelligent and friendly people along the way.

First and foremost, I would like to express my gratitude to my advisor, Noga Alon. His constant support and care have been invaluable; Noga's approach to research, with a sense of humility, a collaborative spirit, and an extremely wide knowledge base, has immensely helped in shaping myself as an aspiring researcher with a desire for cross-disciplinary research. The discussions with Noga throughout the years, be it on interesting research problems or on the academic life, have always been thought-provoking and encouraging.

Eldar Fischer was like a second advisor for me, and I will always be in debt to him for his wise guidance at multiple junctions along my PhD studies. Eldar's insistence on doing research the "right" way without compromising on quality, and his ability to break even the most complex mathematical concepts into simple ideas, have been a source of inspiration to me.

I am thankful to Benny Sudakov for hosting me for a semester at ETH Zurich. Benny's group is the perfect environment for groundbreaking research. I wish to thank Benny and the rest of the group at the time – Igor Balla, Matija Bucić, Nina Kamcev, Matthew Kwan, Shoham Letzter, Alexey Pokrovskiy, and Tuan Tran – for making me feel at home from day one, and for many interesting discussions on research and beyond.

During these last few years I've had the opportunity to collaborate with numerous impressive researchers, many of whom became friends. I would like to thank all of my collaborators throughout the years, and especially Clément Canonne, Daniel Reichman, Danny Hefetz, Erik Waingarten, Eylon Yogev, Joel Oren, Lior Gishboliner, and Simon Korman. Being part of the combinatorics group at Tel Aviv University was a great experience; thanks to Clara Shikhelman, Gal Kronenberg, Peleg Michaeli, Yinon Spinka, Alon Naor, and the rest of the group for enjoyable times. (Playing leg-breaking basketball with you was certainly the best!) I would also like to thank Sarel Cohen for many stimulating conversations.

Embarking on this journey would not have been possible without the encouragement and never-ending support of my family, especially my parents, Doron and Ruth, and my brother Ido. Finally, my deepest admiration goes to my wife, Gal – your endless patience and the ability to lift me up in difficult times have meant the world to me.

Contents

Chapter 1

Introduction

The recent explosion of data-driven approaches in virtually all areas of science and engineering has raised the need to develop efficient algorithms for understanding and analyzing the data, even when one has very limited access to it. The field of *property testing* investigates exactly this topic: what can and cannot be inferred from data given small samples (possibly adaptively crafted, in a data-driven manner). Since its initiation by Rubinfeld and Sudan [122] and by Goldreich, Goldwasser, and Ron [83] around twenty five years ago, this field of study in computer science has enjoyed tremendous progress, stemming both from better mathematical (usually combinatorial, algebraic and topological) understanding of data and what small samples reveal in it, and from remarkable algorithmic achievements in exploiting limited access to data for decision-making, that in many cases have built upon the structural foundations. See e.g. [80, 81, 119, 120] for recent books and surveys.

Formally, the meta-problem in property testing is as follows: we are given query access to the data, represented as an unknown function $f\colon X \to Y$ with known domain X and range Y. The task is to infer efficiently and with good probability whether f satisfies some predetermined property $\mathcal{P}$, or is far from satisfying the property. Given a proximity parameter $\varepsilon > 0$, we say that f is ε-far from $\mathcal{P}$ if one needs to modify it on $\varepsilon|X|$ inputs to make it satisfy $\mathcal{P}$. Efficiency is measured in terms of query complexity, that is, how many queries to the unknown function f one must make in order to complete the above algorithmic task; sometimes the running time is also of interest. Note that since property testing algorithms operate in the sublinear domain, where one cannot even assume access to the whole input, answers are inherently always approximate, and algorithms are always randomized.

Generally, our understanding of property testing is much better when the data has a symmetric, easier-to-exploit structure. This phenomenon was observed and studied in numerous areas of property testing. For example, in probability distribution testing, symmetric parameters (like the entropy of a distribution or its distance to uniformity) are precisely those parameters characterized by the fingerprint – the histogram of the histogram of the sample generated by the distribution –

which in turn led to an excellent understanding of testability in the symmetric regime [135]. In graph property testing (in the dense regime), until recently property testing was well-understood only under the assumption that the property at hand is invariant under vertex-relabeling. In the algebraic front, symmetry and invariances were investigated thoroughly [129], and the general perception is that testing problems become “easier” with more symmetry.

In contrast, for property testing in structures that do not enjoy such inherent symmetry, like sequential data (representing e.g. text, time series data, or biological data), images, and high-dimensional boolean functions, progress has been much slower. Generally, powerful testability results for large families of properties have been much scarcer for these less-symmetric objects, and advances were mostly restricted to a few specific properties of interest, like monotonicity in boolean functions, or convexity and connectivity in images.

In this thesis, we systematically study how to make property testing algorithms “work” when the data is highly structured and does not have any inherent symmetries. The results range from breaking a symmetry barrier in graph property testing, to devising optimal algorithms and lower bounds for detecting global patterns in sequential data, and to the discovery of new generic tools to test any local property in highly structured settings. All problems discussed in this thesis lend themselves to a beautiful combinatorial structure, whose investigation is key to the algorithmic understanding of the problem. Some of the approaches presented here require the development of new combinatorial notions and parameters, which seem to be interesting in their own right.

The thesis is partitioned into three parts. The first part concerns graph property testing in the ordered setting (i.e., without symmetry between the vertices). The second part studies sequential data from the property testing perspective, specifically the problem of detecting global structured patterns in sequential data. In the third part we investigate the testability of local properties in the structured regime. For each of these parts, we now describe the problem we address and its background, the main algorithmic results, and the combinatorial ideas behind these results.

We henceforth assume that the reader is familiar with standard property testing notation and definitions; see Section 1.4 for the relevant notation.

1.1 Ordered Graphs: Regularity and Removal

The first part of this thesis (Chapter 2) develops removal lemmas (and, as a byproduct, very general testability results) for graphs and matrices without any symmetry requirement. A long line of work in property testing focused on characterizing the efficiently testable unordered graph properties, which by definition are closed under relabeling of the vertices. However, these works do not address the case where vertex labels are important, e.g., in ordered graphs or images (which can be viewed as ordered bipartite graphs, where the locations of pixels, and so of rows and columns of the representing matrix, are meaningful). All results mentioned here are for the *dense* graph model, where a graph is represnted by a function $G\colon \binom{[n]}{2} \to \{0,1\}$.

In the seminal paper of Goldreich, Goldwasser and Ron [83] it was shown that all unordered graph properties that can be represented by a certain graph partitioning, including properties such as k-colorability and having a large clique, are testable with a constant number of queries (see also [86]). Alon, Fischer, Krivelevich and Szegedy [7] proved that the unordered property of $\mathcal{F}$-freeness, i.e., of not containing as an induced subgraph any $F \in \mathcal{F}$ (ignoring the vertex labels) is testable with a constant number of queries for any finite family $\mathcal{F}$ of forbidden graphs. Their result follows directly from a graph-theoretic statement, the *induced graph removal lemma*, which is a generalization of the well-known *graph removal lemma* of Ruzsa and Szemerédi [5, 131], and states the following. For any finite family $\mathcal{F}$ of unordered graphs and $\varepsilon > 0$ there exists $q = q(\mathcal{F}, \varepsilon) > 0$, such that for any graph G which is ε-far from $\mathcal{F}$-freeness, a random induced subgraph of G on q nodes contains an induced copy of some $F \in \mathcal{F}$ with probability at least $2/3$. The proof uses a strengthening of the celebrated Szemerédi graph regularity lemma [131], known as the *strong graph regularity lemma.*

Proving testability results for graphs directly from removal lemmas, which in turn are proved using regularity lemmas, has since become the go-to method for exploring testability in graphs (and for good reason; later it was shown [9] that constant-query testability and regularity are equivalent in a rather strong sense). Alon and Shapira [12] generalized the induced graph removal lemma of [7] to infinite families. Since any *hereditary* graph property $\mathcal{P}$ (i.e., any property of unordered graphs that is closed under deletion of vertices) is characterized by a family $\mathcal{F}$ (finite or infinite) of forbidden induced subgraphs, these results imply a remarkably general testability result.

Theorem ([12]). *Any hereditary property of unordered graphs is constant-query testable.*

An efficient finite induced removal lemma for binary unordered matrices, with no row and column order, was obtained by Alon, Fischer and Newman [8] in 2007. Here, a matrix is a function $M\colon [n] \times [m] \to \Sigma$, where Σ is some fixed alphabet (say, $|\Sigma| = 2$ corresponds to a binary matrix). An $s \times t$ submatrix is any restriction of M to specific s rows and t columns (not necessarily consecutive), where the order of the rows and columns is inherited from M. The main tool in [8] is an efficient conditional regularity lemma for ordered binary matrices, and it was conjectured there that this regularity lemma can be used to obtain a removal lemma for ordered binary matrices. Again, by *ordered* here we mean that, unlike all previously mentioned results, labels in the forbidden submatrix *are* important: being free of a forbidden family of ordered (labeled) materices $\mathcal{F}$ amounts to not containing any of them as an isomorphic copy *with the same row and column order.* Thus, there is no symmetry between the labels that we can utilize.

Conjecture (Ordered matrix removal lemma [8]). *For any finite family $\mathcal{F}$ of ordered binary matrices and any $\varepsilon > 0$ there exists $\delta = \delta(\mathcal{F}, \varepsilon)$ such that any $n \times n$ binary matrix which is ε-far from $\mathcal{F}$-freeness contains at least δn^{a+b} copies of some $a \times b$ matrix from $\mathcal{F}$.*

The main result in the first part is a proof of this conjecture from 2007, which in fact holds for edge-colored graph and matrices over any fixed alphabet, and unlike all previous results for graphs, does not assume any symmetry condition on the labels.

Theorem. *Any hereditary property of graphs or matrices (over a fixed alphabet) satisfies a removal lemma, and is thus constant-query testable; this holds for both ordered and unordered properties.*

Technical Foundations. Proving graph removal lemmas (and, consequently, testability results for hereditary graph properties) typically goes through the definition of a suitable "regularity schemes": a simplified, constant size structure that satisfies two crucial properties. The first of them is closeness to the original graph, that is, one can make the original graph identical to a blowup of the regularity scheme with only a very small number of modifications. On the other hand, the scheme should be representative enough of the original graph, in the sense that any substructure found in it must also be abundant in the original graph.

For the most basic (non-induced) graph removal lemma, the regularity scheme is just the regular partition given by Szemerédi regularity lemma [131]. In the more complicated case of the induced graph removal lemma (as well as in its infinite analogue), more complicated, nested regularity schemes are required. However, even these schemes fail to capture order. The main technical contribution here is the development of substantially more sophisticated schemes that take order into account. It is shown how to construct a scheme that provides both order-regularity (extending order-regularity notions for sequences) and graph-regularity simultaneously. Along the way, and in order to address both order- and graph-regularity, a new type of Ramsey-type theorem with *undesirable edges* in multipartite graphs is developed. Roughly speaking, this theorem states that any sufficiently large multipartite graph contains a substructure which is monochromatic between each two parts, while also not having many undesirable edges in the structure.

References. The results of this chapter appear in:

- N. Alon, O. Ben-Eliezer, E. Fischer, *Testing hereditary properties of ordered graphs and matrices*, Proc. 58th Annual IEEE Symposium on Foundations of Computer Science (*FOCS*), 2017, 848–858.

1.2 Property Testing Algorithms for Sequential Pattern Detection

In the second part of the thesis (Chapters 3-5), we consider a structured, massively parameterized property testing problem in sequences $f : [n] \to \mathbb{R}$, raised and first studied by Newman, Rabinovich, Rajendraprasad, and Sohler [108, 109]. Fix a permutation[1] $\pi \colon [k] \to [k]$. A sequence $f \colon [n] \to \mathbb{R}$ contains π as an *order pattern* if there exist $i_1 < i_2 < \ldots < i_k$ such that $f(i_j) < f(i_{j'})$ if and only if $j, j' \in [k]$ satisfy that $\pi(i_j) < \pi(i_{j'})$. That is, intuitively, if we consider the subsequence of f in

[1]Despite calling π a permutation here, we view it as a combinatorial object rather than a group-theoretic one.

locations $i_1 < \ldots < i_k$ and only look at the relative order of these values, this is the exact same structure as in π. Considering this from the property testing perspective, Newman et al. initiated the study of testing *order pattern freeness*, i.e., the property of not containing a fixed known pattern π. The focus is on the regime where k and the proximity parameter ε are constant, and on one-sided error tests, where to deduce that a sequence is not pattern-free, the test must prove the existence of a pattern. This problem is naturally motivated by data-series analysis, where the central task is to efficiently identify global patterns in massive-scale sequential data, and is closely related to other classical problems in sequences, like the estimation of the longest increasing subsequence (LIS). The simplest special case of the problem, where $\pi = (2, 1)$, corresponds to perhaps the most well-investigated property in the testing literature, *monotonicity testing* (see Section 3.1 for an extensive background).

In their paper [109], Newman et al. proved several results of interest, hinting that this problem underlies rich combinatorial phenomena, highly dependent on the structure of the pattern π and the adaptivity of the property testing algorithm. With regards to *structure*, it was shown that π-freeness is very efficiently testable non-adaptively if and only if π is monotone: on the one hand, if $\pi = (1, 2, \ldots, k)$ or $\pi = (k, k-1, \ldots, 1)$ then π-freeness is testable non-adaptively with $(\log n)^{O(k^2)}$ non-adaptive queries. On the other hand, any non-monotone pattern requires $\Omega(\sqrt{n})$ queries - an exponential separation. This is shown to be near-tight for $\pi = (1, 3, 2)$. From the adaptivity perspective, it was shown that again there is an exponential separation: compared with the $\Omega(\sqrt{n})$ non-adaptive lower bound, the pattern $\pi = (1, 3, 2)$ requires only a polylogarithmic number of *adaptive* queries. In summary, the results of Newman et al. reveal that both the structure of the pattern π (i.e., monotone vs. non-monotone) and our ability to react adaptively matter a lot in this problem. However, the general task of understanding the query complexity of optimal tests for π-freeness – for any π – both in the adaptive and the non-adaptive case, has remained wide open. The major open problems that they pose are the following.

1. *Adaptive case.* Is it true that π-freeness is testable adaptively with query complexity polylogarithmic in n for *any* permutation π?

2. *Non-adaptive case.* How does the structure of a pattern π correlate with the query complexity of an optimal non-adaptive test for π-freeness? In particular, are there infinitely many non-monotone permutations π for which π-freeness is testable with query complexity $O(n^{0.99})$?

In this thesis, we make progress in the understanding of both the adaptive and the non-adaptive front. While the first open question above seems very difficult, we settle the second question. Together with the work of Newman et al., the results presented here provide an essentially full understanding of monotone patterns (both the adaptive and the non-adaptive case) for fixed ε and k, as well as good (but not yet complete) understanding of the power of non-adaptive tests. The results shed light on multiple intriguing combinatorial parameters that arise naturally in the study of this problem (and to the best of our knowledge, did not appear before in the combinatorial literature), which seem very interesting on their own right.

1.2.1 Monotone Patterns

The results of Newman et al. [109] show that for any monotone pattern $\pi = (1, 2, \ldots, k)$, the property of order pattern freeness paramtrized by π is testable with a polylogiathmic number of queries, $(\log n)^{O(k^2)}$, for fixed k and ε (which will in general be our domain of interest here). But what is the *correct* polylogarithmic dependence here?

While the authors of [109] did not try to optimize the above dependence, their approach yields, in principal, a query complexity of $(\log n)^{O(k)}$. Their main observation is the decomposability of this testing problem when π is monotone: Namely, if we concatenate two monotone subsequnces of f of lengths $\ell, k - \ell$, where the starting point of the second subsequence is both “higher” (in terms of value) and “further to the right” (in terms of location) than the last element of the first subsequence, then together they form a monotone subsequence of length k. This property allows one to test $(1, 2, \ldots, k)$-freeness recursively (though non-adaptively; namely, the collection of indices to be queried is defined recursively). By carefully unrolling the recursion and enumerating over possible “widths” of copies (i.e., the typical distance between the end of each copy and its beginning) on a logarithmic scale, one can obtain the $(\log n)^{O(k)}$ upper bound. On the other hand, a lower bound of $\Omega(\log n)$, for both adaptive and non-adaptive algorithms, follows from monotonicity testing lower bounds [63, 66], i.e. the case $k = 2$, for which it is tight. Between $(\log n)^{O(k)}$ and $\Omega(\log n)$, what would be the query complexity for non-adaptive algorithms? does adaptivity help?

We resolve these two questions here. Strikingly, the answer to the first question is precisely $\Theta((\log n)^{\lfloor \log_2 k \rfloor})$ for any fixed k and ε. The answer to the second question is positive, and in fact, for any fixed k, only $O(\log n)$ queries are required. That is, despite the fact that testing $(1, 2, \ldots, k)$-freeness is structurally much more complicated to analyze than monotonicity testing, and the fact that there is a non-adaptive separation between these two properties when $k \geq 4$, for adaptive algorithms the query complexity in both tasks is of the same order of magnitude for *every* fixed k.

Splittable Intervals and Growing Suffixes. The main building block for the new results on testing monotone patterns is a surprisingly powerful characterization of sequences that are far from $(1, 2, \ldots, k)$-freeness, described in detail in Chapter 3. This is a “structure versus chaos” type characterization (see e.g. the book of Tao [133] on such characterizations), stating that either most disjoint copies of $(1, 2, \ldots, k)$ have one out of constantly many possible widths (the “structure” case, named *splittable intervals*), or these copies are interwoven enough to organically form many long and easy-to-detect monotone subsequences in the data (the “chaos” case, named *growing suffixes*).

Non-Adaptive results for monotone patterns. The lower bound of $\Omega(\log n)$ for monotonicity testing constructs $\log n$ examples, each with a different "distance profile", where queries that are helpful in determining whether a sequence has any particular profile are completely useless for extracting information about other profiles. This construction can be iterated: for the pattern $\pi = (1, 2, \ldots, k)$, one can construct $\binom{\log n}{\log k}$ different profiles in a recursive fashion. The key observation is that with q queries, one can only capture $q-1$ different iterated profiles, which establishes the non-adaptive lower bound. The more interesting direction is algorithmic, and shows the $O((\log n)^{\lfloor \log_2 k \rfloor})$ upper bound. In the growing suffixes (i.e., chaotic) case, monotone subsequences are easy to find and this case equires just $O(\log n)$ queries for any fixed k. The splittable intervals case requires substantially more work; the main idea here is to repeatedly apply this condition to construct a "tree profile" with k nodes, which describes the typical structure of $(1, 2, \ldots, k)$-copies in the sequence. The key algorithmic observation is that with only $O(\log n)$ queries, one can shatter any tree profile, non-adaptively, in a way that all the remaining subtrees are of size at most $k/2$. Applying this argument inductively leads to the $O((\log n)^{\lfloor \log_2 k \rfloor})$ upper bound. See Chapter 3.

Adaptive results for monotone patterns. The algorithmic approach of the non-adaptive case, which enumerates over all tree profiles, cannot carry on to the adaptive setting. Indeed, all previous methods enumerate over all possible widths at any given depth of the recursion, incurring a $O(\log n)$ multiplicative price at any such depth.

Instead, the approach here is based on "wishful thinking". One makes only $O(\log n)$ queries, and picks a couple of elements believed to be the endpoints (i.e., the 1- and k-entries) of some $(1, 2, \ldots, k)$-copy. It is shown that with good probability, either this couple of elements indeed forms the endpoints of a copy (the *hitting* case), or they are much too wide to be such endpoints (the *overshooting* case). The key observation, combining a strengthened form of the aforementioned growing suffixes vs. splittable intervals characterization with additional combinatorial ideas. shows that surprisingly, the overshooting case can also be handled efficiently. To the best of our knowledge, these combinatorial ideas were not used before in the property testing literature. See Chapter 4 for more details.

References. The results of these chapters appear in:

- O. Ben-Eliezer, C. Canonne, S. Letzter, E. Waingarten, *Finding monotone patterns in sublinear time*, Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (*FOCS*), 2019, 1457–1482.

- O. Ben-Eliezer, S. Letzter, E. Waingarten, *Optimal adaptive detection of monotone patterns.*

1.2.2 Non-Monotone Patterns

The results of Newman et al. [109] indicate that for some patterns, even sophisticated non-adaptive algorithms are almost useless. Specifically, while most naive test, based on uniform sampling, requires $O(n^{1-1/k})$ queries to locate a pattern, there exist patterns which require $\Omega(n^{1-2/(k+1)})$ queries. This thesis substantially extends the understanding of the non-adaptive regime. On the positive side, it is shown that any π can be found using $O(n^{1-1/(k-1)})$ queries. The algorithm combines random samples with querying of consecutive intervals of elements. Conversely, for any fixed k there exist patterns with a matching $\Omega(n^{1-1/(k-1)})$ lower bound on the query complexity. This phenomenon is in fact much more general: for *almost all* patterns of length k (a fraction that tends to 1 as $k \to \infty$) there is a non-adaptive lower bound of $\Omega(n^{1-1/(k-3)})$, thereby establishing that the non-adaptive complexity of all such patterns is $n^{1-1/(k-\Theta(1))}$. In other words, the main message here is that sophisticated non-adaptive methods are almost useless, they are barely sublinear, and improve upon uniform sampling very marginally. Additionally, a structural hierarchy theorem is proved, which shows that for any $1 \leq \ell \leq k-1$, there are patterns whose non-adaptive complexity is $\tilde{\Theta}(n^{1-1/\ell})$. This settles the second part of the second open question above, from [109].

From the combinatorial perspective, the lower bounds proved here seem closely related to a combinatorial parameter of permutations, the *unique signed partition number*, which has not been defined and investigated before; we conjecture that in fact, this parameter controls the non-adaptive query complexity. The parameter is rather complicated and in this introduction we present a simpler and more elegant parameter, the *stitching number*, which is of the same spirit, and whose investigation implies the aforementioned general lower bound for almost all length-k patterns. Given a pattern π with $|\pi| = k$ and where (assuming without loss of generality that $\pi^{-1}(1) < \pi^{-1}(k)$) the stitching number $s(\pi)$ of π is the number of pairs of neighboring elements one needs to stitch together, so that the union of these stitched pairs satisfies the following:

1. For all pairs (a, b) in the stitching, $a < b$ and $\pi^{-1}(b) = \pi^{-1}(a) + 1$.

2. For any element $i \in [k]$, there exists a pair (a, b) in the stitching where $a \leq i \leq b$.

It is not hard to verify that almost all patterns of length k have stitching number 2 or 3. Our lower bounds imply that any pattern π requires $\Omega(n^{1-1/(k-s(\pi))})$, thereby proving the aforementioned general lower bound. Moreover, the hardest-to-test patterns are those where $s(\pi) = 1$, that is, 1 and k are neighbors. See Chapter 5 for more details.

References. The results of this chapter appear in:

- O. Ben-Eliezer, C. Canonne, *Improved bounds for testing forbidden order patterns*, Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (*SODA*), 2018, 2093–2112.

1.3 Understanding Locality in Structured Property Testing

In the third part of the thesis, Chapters 6-7, we consider *local* properties in multi-dimensional arrays. A d-dimensional array over the alphabet Σ is a function $A\colon [n]^d \to \Sigma$. In the one dimensional case, a property is considered k-local if it can be defined by a collection of forbidden consecutive substrings of length k. For example, the property of monotonicity is 2-local: an array is non-decreasing if and only if there is no pair of consecutive elements in A that are decreasing, that is, no i such that $A(i) > A(i+1)$. More generally, a property of d-dimensional arrays is k-*local* if it can be defined by a collection $\mathcal{F}$ of "forbidden" consecutive d-dimensional patterns of size $k \times k \times \ldots \times k$. That is, A satisfies the local property defined by $\mathcal{F}$ if for any $(a_1, \ldots, a_d) \in [n-k]^d$, there is no $F \in \mathcal{F}$ satisfying $A(a_1 + x_1, \ldots, a_d + x_d) = F(x_1, \ldots, x_d)$ for all $(x_1, \ldots, x_d) \in [k]^d$.

Some of the most well-investigated and interesting properties in the testing literature (and many properties never investigated from the property testing perspective) are local. Monotonicity and Lipschitz continuity are 2-local for any d. (Discrete) convexity is usually 3- or 4-local (depending on the definition). More generally, properties defined by discrete derivatives of order k are $(k+1)$-local. Submodularity is 2-local. Many problems in more applied areas, like computational biology or computer vision, are k-local for small k. See Section 6.1 for a definition and discussion of these properties.

Despite being relatively natural and capturing a wide range of properties, the above definition of locality has never been formally defined and analyzed in the literature. In this part, general testability results are proved for all local properties. Furthermore, improved bounds are obtained for the property of pattern-freeness, where F consists of a single forbidden pattern, through a combinatorial analysis of a pattern deletion problem.

1.3.1 Testing Local Properties: Follow the Boundary

The main result of Chapter 6 is a generic one-sided error test for all k-local properties in d-dimensional arrays over any finite (but not necessarily bounded-size) alphabet Σ. The query complexity of the test is $O(\frac{k}{\varepsilon} \cdot \log \frac{\varepsilon n}{k})$ for $d = 1$ and $\frac{k}{\varepsilon^{1/d}} \cdot (O(n))^{d-1}$ for $d > 1$. When k and ε are fixed, these expressions are $O(\log n)$ and $O(n^{d-1})$, respectively. The results imply, in particular, that any property with locality $o(n)$ is sublinearly testable. They are tight in various cases, both in one dimension and in higher dimensions.

Interestingly, the generic test proposed here gives the first sublinear-query test for two challenging properties in high dimensions, convexity and submodularity. The query complexity for both is $O(n^{d-1})$. This, obviously, leaves much to be desired. Subsequent work by Belovs, Blais, and Bommireddi [20] shows that to get a significant improvement for convexity testing, one would have to understand how to use adaptivity when testing these local properties. Specifically, without adaptivity, the $O(n)$ query complexity of the generic test is optimal for two-dimensional convexity, and for higher fixed d, there is a lower bound of $\Omega((n/d)^{d/2})$ for testing d-dimensional convexity with non-adaptive algorithms.

Technical Foundations. Let us revisit the original proof that monotonicity is testable with $O(\log n)$ queries, by Ergün et al. [63]. Pick $i \in [n]$ uniformly at random, and consider an imaginary binary search in $[n]$, starting in the middle point of the array and culminating in i. Let $Q \subseteq [n]$ be the set of all visited points in the binary search. The main claim of [63] is that if a sequence is ε-far from monotonicity, then with probability at least ε, querying Q will reveal a violation to monotonicity and lead to rejection.

A striking feature of the above idea is that it (essentially, with suitable extensions) can be applied to any local property $\mathcal{P}$. For simplicity, let us focus on the simplest case $d = 1$ and $k = 2$. We say that a consecutive subarray S of an array A is *unrepairable* if, when fixing the values on the boundaries of the subarray S (i.e., its first and last element) but allowing one to change the values in the interior of S arbitrarily, the modified S will never satisfy $\mathcal{P}$. For monotonicity, this notion is rather natural: S is unrepairable if and only if its first element is larger than the last one. However, as it turns out, the above test for monotonicity actually works for *any* local property, with the only change being that instead of verifying whether pairs of queried elements do not violate monotonicity, we check whether these pairs are unrepairable.

References. The results of this chapter appear in:

- O. Ben-Eliezer, *Testing local properties of arrays*, Proc. 10th Innovations in Theoretical Computer Science (*ITCS*), 2019, 11:1–11:20.

1.3.2 Testing Meets Pattern Matching: the Modification Lemma

The generality of the above test makes it, naturally, sub-optimal for many properties of interest. One of them is the property of P-freeness for a single forbidden pattern P, studied in Chapter 7. The motivation for studying this property stems from pattern matching applications in computational biology, computer vision, and other areas, where the fast detection of local patterns is among the most important algorithmic tasks. The main result here is a tolerant test for pattern freeness (for large enough patterns), whose query complexity depends only on the dimensionality d and (inversely linearly) on ε. Crucially, unlike the above generic test, the complexity is independent of n. In one dimension, it is shown that computing the *exact distance* of an array A from P-freeness can be done in linear time (in high dimensions, however, exactly computing the distance is NP-hard [72]). Indeed, there exists an explicit "hitting set" of entries in A, whose modification simultaneously deletes all existing P-copies in A, without creating any new copies of P in A. That is, the hitting number of the copies in P is equal to the distance from freeness. This observation seems useful in computational biology, where it is required to clean genetic sequences from undesirable patterns with as few modifications as possible; we leave this as an open problem.

Technical Foundations. The combinatorial core of the proof, which we call a *modification lemma* (as it is somewhat similar in spirit to removal lemmas), states the following: for almost any large enough d-dimensional pattern P, and any array A containing a copy of P, there exists an entry within the copy with the following property: if we change the value of this entry, the copy of P will obviously be destroyed, but additionally, no new copies of P will be created. Such a modification lemma leads naturally to a removal lemma: it means that the number of P-copies in A is at least its distance from P-freeness.

References. The results of this chapter appear in:

- O. Ben-Eliezer, S. Korman, D. Reichman, *Deleting and testing forbidden patterns in multi-dimensional arrays*, Proc. 44th International Colloquium on Automata, Languages and Programming (*ICALP*), 2017, 9:1–9:14.

1.4 Notation

The general setting in property testing is as follows. We are given query access to an unknown function $f\colon X \to Y$ where the domain X and the range Y are known. A query is the operation of obtaining the value of $f(x)$ for $x \in X$ of our choice. Given a property $\mathcal{P}$ (a family of functions from X to Y), and a proximity parameter $\varepsilon > 0$, we say that f is ε-far from $\mathcal{P}$ if any function $g \in \mathcal{P}$ differs from f on at least $\varepsilon|X|$ inputs. if f is not ε-far from $\mathcal{P}$, then we say that it is ε-close to $\mathcal{P}$. The algorithmic task in property testing is to decide, with success probability at least (say) 2/3, whether f satisfies $\mathcal{P}$ or is ε-far from $\mathcal{P}$. Here $\mathcal{P}$ and ε are generally known in advance.[2] An algorithm for the above task is called an ε-test. There are also other variants of the testing task, such as tolerant testing, where the algorithm is required to distinugish between ε_1-closeness to $\mathcal{P}$ and ε_2-farness from $\mathcal{P}$, for some $0 \le \varepsilon_1 < \varepsilon_2$. Many of the algorithms discussed in this thesis are *non-adaptive*; by this we mean that all queries are chosen in advance, before seeing any of the values of f. Any algorithm whose decisions depend in some way on the results of queries it makes is considered *adaptive*.

A property testing algorithm which, for $f \in \mathcal{P}$ always (with probability 1) decides correctly that f indeed satisfies $\mathcal{P}$, is said to have *one-sided error*. Any testing algorithm that does not satisfy this condition has *two-sided error*.

Along the thesis we use several standard notions. $[n]$ denotes the set $\{1, 2, \ldots, n\}$; the notion of $\tilde{\Theta}(\cdot)$ will usually refer to an expression that hides terms that are polylogarithmic in n (but not always; we will indicate whenever this is not the case). We sometimes omit floor and ceiling signs when they are not essential; note that one of the results in this thesis has a floor sign in

[2]Another model for property testing, *proximity oblivious testing*, does not assume that ε is known, and instead focuses on designing basic testing algorithms with small success probability, which can be amplified by repeating the basic test many times. See e.g. the book of Goldreich [81] for an extensive discussion.

the exponent, which *is* essential and included wherever relevant. Logarithms are in base 2, unless stated otherwise. We will, at times, suppress polynomial factors by writing $\text{poly}(\cdot)$ to refer to a large enough polynomial in the relevant parameter, whose degree is a large enough universal constant. Terms such as $O_k(\cdot)$ and $\Omega_k(\cdot)$ are similar to $O(\cdot)$ and $\Omega(\cdot)$ except that the underlying constants may depend on the parameter k.

Part I

Ordered Graphs: Regularity and Removal

Chapter 2

Removal Lemma for Ordered Graphs and Matrices

The results in this chapter appear in [4].

We focus here on property testing of two-dimensional structures over a fixed finite alphabet Σ, in particular graphs and matrices, and start with some notation. *Graphs* are functions $G\colon \binom{V}{2} \to \{0,1\}$ where V is the vertex set; more generally *edge-colored graphs* (with fixed finite color set Σ) are functions $G\colon \binom{V}{2} \to \Sigma$. *Matrices* over the alphabet Σ (or *images*) are functions $M\colon U \times V \to \Sigma$. We generally consider edge-colored graphs rather than standard graphs, as the added generality will prove useful later; the term *graph* usually refers to an edge-colored graph. Thus, any collection of (edge-colored) graphs $G\colon \binom{V}{2} \to \Sigma$ is an *ordered graph property.* As a special case, an *unordered graph property* is an ordered graph property that is also *invariant under vertex permutations*: If $G \in \mathcal{P}$ and π is any permutation on V_G, then the graph G^π, defined by $G^\pi(\pi(u)\pi(v)) = G(uv)$ for any $u \neq v \in V_G$, satisfies $G^\pi \in \mathcal{P}$. Similarly, an *(ordered) matrix property*, or an *image property*, is a collection of functions $M\colon [m] \times [n] \to \Sigma$. We generally assume here (unless it is explicitly stated that we consider unordered graphs) that the vertex set V of a graph G has a total ordering (e.g. the natural one for $V = [n]$), which we denote by $<$. The (induced) *ordered subgraph* of the graph $G : \binom{V}{2} \to \Sigma$ on $U \subseteq V$, where the elements of U are $u_1 < \ldots < u_k$, is the graph $H\colon \binom{[k]}{2} \to \Sigma$ which satisfies $H(ij) = G(u_i u_j)$ for any $i < j \in [k]$. For a family $\mathcal{F}$ of "forbidden" graphs, the property $\mathcal{P}_\mathcal{F}$ of *$\mathcal{F}$-freeness* consists of all graphs G for which any ordered subgraph H of G satisfies $H \notin \mathcal{F}$. Finally, a property $\mathcal{P}$ is *hereditary* if it is closed under taking induced subgraphs. That is, for any $G \in \mathcal{P}$ and ordered subgraph H of G, it holds that $H \in \mathcal{P}$. Note that a property $\mathcal{P}$ is hereditary if and only if $\mathcal{P} = \mathcal{P}_\mathcal{F}$ for some (finite or infinite) family $\mathcal{F}$ of graphs over Σ. The analogous notions of ordered subgraphs, $\mathcal{F}$-freeness and hereditary properties for matrices are "structure preserving". Here, the *ordered submatrix* of the matrix $M : [m] \times [n] \to \Sigma$ on $A \times B$, where the elements of A and B are $a_1 < \ldots < a_k$ and $b_1 < \ldots < b_l$, is the matrix $N : [k] \times [l] \to \Sigma$ defined by $N(i,j) = M(a_i, b_j)$ for any $i \in [k]$ and $j \in [l]$.

2.1 Introduction

Some of the most interesting results in property testing have been those that identify large families of properties that are efficiently testable, and those that show that large families of properties cannot be tested efficiently. One of the most widely investigated questions in property testing has been that of characterizing the efficiently testable unordered graph properties. In the seminal paper of Goldreich, Goldwasser and Ron [83] it was shown that all unordered graph properties that can be represented by a certain graph partitioning, including properties such as k-colorability and having a large clique, are constant-query testable (see also [86]). Alon, Fischer, Krivelevich and Szegedy [7] showed that the property of $\mathcal{F}$-freeness is constant-query testable for any finite family $\mathcal{F}$ of forbidden unordered graphs (here the term *unordered graphs* refers to the usual notion of graphs with no order on the vertices). Their main technical result, now known as the *induced graph removal lemma*, is a generalization of the well-known *graph removal lemma* [5, 124].

Theorem 2.1 (Induced graph removal lemma [7]). *For any finite family $\mathcal{F}$ of unordered graphs and $\varepsilon > 0$ there exists $\delta = \delta(\mathcal{F}, \varepsilon) > 0$, such that any graph G which is ε-far from $\mathcal{F}$-freeness contains at least δn^q copies of some $F \in \mathcal{F}$ with q vertices.*

The original proof of Theorem 2.1 uses a strengthening of the celebrated Szemerédi graph regularity lemma [131], known as the *strong graph regularity lemma.*

It is clear that having a removal lemma for a family $\mathcal{F}$ immediately implies that $\mathcal{F}$-freeness is constant-query testable: A simple test which picks a subgraph H whose size depends only on $\mathcal{F}$ and ε, and checks whether H contains graphs from $\mathcal{F}$ or not, is a valid one-sided test for $\mathcal{F}$-freeness. Hence, removal lemmas have a major role in property testing. They also have implications in different areas of mathematics, such as number theory and discrete geometry. For more details, see the survey of Conlon and Fox [56].

By proving a variant of the induced graph removal lemma that also holds for infinite families, Alon and Shapira [12] generalized the results of [7]. The infinite variant is as follows.

Theorem 2.2 (Infinite graph removal lemma [12]). *For any finite or infinite family $\mathcal{F}$ of unordered graphs and $\varepsilon > 0$ there exist $\delta = \delta(\mathcal{F}, \varepsilon) > 0$ and $q_0 = q_0(\mathcal{F}, \varepsilon)$, such that any graph G which is ε-far from $\mathcal{F}$-freeness contains at least δn^q copies of some $F \in \mathcal{F}$ on $q \leq q_0$ vertices.*

Theorem 2.2 implies that *any* hereditary unordered graph property is constant-query testable, exhibiting the remarkable strength of Szemerédi regularity based approaches for property testing.

Theorem 2.3 (Hereditary graph properties are constant-query testable [12]). *Let Σ be a finite set with $|\Sigma| \geq 2$. Any hereditary unordered graph property over Σ is constant-query testable.*

Alon, Fischer, Newman and Shapira later presented [9] a complete combinatorial characterization of the graph properties that are testable (with two-sided error) using a constant number of queries, building on results from [70, 86]. Independently, Borgs, Chayes, Lovász, Sós, Szegedy and

Vesztergombi obtained an analytic characterization of such properties through the theory of graph limits [37]. See also [101, 102].

An efficient finite induced removal lemma for binary matrices with no row and column order was obtained by Alon, Fischer and Newman [8]. In this case, δ^{-1} is polynomial in ε^{-1} (where ε, δ play the same roles as in the above removal lemmas). It was later shown by Fischer and Rozenberg [71] that when the alphabet is bigger than binary, the dependence of δ^{-1} on ε^{-1} is super-polynomial in general. Actually, the main tool in [8] is an efficient conditional regularity lemma for ordered binary matrices, and it was conjectured there that this regularity lemma can be used to obtain a removal lemma for ordered binary matrices.

Conjecture 2.4 (Ordered binary matrix removal lemma [8]). *For any finite family $\mathcal{F}$ of ordered binary matrices and any $\varepsilon > 0$ there exists $\delta = \delta(\mathcal{F}, \varepsilon)$ such that any $n \times n$ binary matrix which is ε-far from $\mathcal{F}$-freeness contains at least δn^{a+b} copies of some $a \times b$ matrix from $\mathcal{F}$.*

In contrast to the abundance of general testing results for two-dimensional structures with an inherent symmetry, such as unordered graphs and matrices, no similar results for *ordered* two-dimensional structures (i.e. structures that do not have any underlying symmetry) have been established. Even seemingly simple special cases, such as F-freeness for a single ordered graph F, or M-freeness for a single 2×2 ordered matrix M, are not known to be constant-query testable in general [6]. A good survey on the role of symmetry in property testing is given by Sudan [129], who suggests that the successful characterization of the constant-query testable unordered graph properties is attributable to the underlying symmetry of these properties; See also [84].

Despite the lack of general results as above for the ordered case, property testing of multi-dimensional ordered structures has recently been an active area of research. This is discussed in the third part of this thesis. Ordered graphs were less investigated in the context of property testing, but are the subject of many works in Combinatorics and other areas. See, e.g., a recent work on Ramsey-type questions in the ordered setting [57], in which it is shown that Ramsey numbers of simple ordered structures might differ significantly from their unordered counterparts.

Finally, we mention a relevant result on *one-dimensional* structures. Alon, Krivelevich, Newman and Szegedy [10] showed that regular languages are constant-query testable. One can combine this result with the well-known Higman's lemma in order theory [91] to show that any hereditary property of words (i.e. one dimensional functions) over a finite alphabet is constant-query testable.

Our contributions

We prove generalizations of Theorems 2.3 and 2.2 to the ordered setting, as well as analogous results for matrices. The following result generalizes Theorem 2.3.

Theorem 2.5 (Hereditary properties of ordered graphs are constant-query testable). *Fix a finite set Σ with $|\Sigma| \geq 2$. Any hereditary ordered graph property over Σ is constant-query testable.*

To prove Theorem 2.5, we establish an *order-preserving* induced graph removal lemma, which holds for finite and infinite families of ordered graphs. This is a generalization of Theorem 2.2.

Theorem 2.6 (Infinite ordered graph removal lemma). *Fix a finite set Σ with $|\Sigma| \geq 2$. For any (finite or infinite) family $\mathcal{F}$ of ordered graphs $F : \binom{[n_F]}{2} \to \Sigma$ and any $\varepsilon > 0$ there exist $q_0 = q_0(\mathcal{F}, \varepsilon)$ and $\delta = \delta(\mathcal{F}, \varepsilon) > 0$, such that any ordered graph $G : \binom{[n]}{2} \to \Sigma$ that is ε-far from $\mathcal{F}$-freeness contains at least δn^q induced copies of some graph $F \in \mathcal{F}$ on $q \leq q_0$ vertices.*

An analogue of Theorem 2.5 for matrices is also proved.

Theorem 2.7 (Hereditary properties of ordered matrices are constant-query testable). *Fix a finite set Σ with $|\Sigma| \geq 2$. Any hereditary (ordered) matrix property over Σ is constant-query testable.*

As in the case of ordered graphs, to prove Theorem 2.7 we establish the following ordered matrix removal lemma, which holds for finite and infinite families of matrices, and settles a generalized form of Conjecture 2.4.

Theorem 2.8 (Infinite ordered matrix removal lemma). *Fix a finite set Σ with $|\Sigma| \geq 2$. For any (finite or infinite) family $\mathcal{F}$ of ordered matrices over Σ and any $\varepsilon > 0$ there exist $q_0 = q_0(\mathcal{F}, \varepsilon) > 0$ and $\delta = \delta(\mathcal{F}, \varepsilon) > 0$, such that any ordered matrix over Σ that is ε-far from $\mathcal{F}$-freeness contains at least $\delta n^{q+q'}$ copies of some $q \times q'$ matrix $F \in \mathcal{F}$, where $q, q' \leq q_0$.*

Actually, the proof of Theorem 2.8 is almost identical to that of Theorem 2.6, so we only describe what modifications are needed to make the proof of Theorem 2.6 also work here, for the case of square matrices. However, all proofs can be adapted to the non-square case as well. An outline for the proof of the graph case is given in Section 2.2, and all of the sections after it are dedicated to the full proof. The needed modifications for the matrix case appear in Section 2.6.3.

To the best of our knowledge, Theorems 2.5 and 2.7 are the first known testing results of this type for ordered two-dimensional structures, and Theorems 2.6 and 2.8 are the first known order-preserving removal lemmas for two-dimensional structures.

The author and Fischer [24] showed that for any property $\mathcal{P}$ of ordered graphs and matrices which is constant-query testable using a canonical test, which picks uniformly at random vertices and queries the subgraph (or submatrix) induced over these vertices, one can also tolerantly test $\mathcal{P}$, or equivalently, estimate with probability $2/3$ the distance of the input to $\mathcal{P}$, with a constant number of queries. That is, we have the following corollary.

Corollary 2.9 (see also [24]). *Any hereditary property $\mathcal{P}$ of ordered graphs and matrices is tolerantly testable with a constant number of queries. That is, for any $0 \leq \alpha < 1$, the property of α-closeness to $\mathcal{P}$ is constant-query testable.*

It is interesting to note that some properties previously investigated in the literature, such as monotonicity, k-monotonicity [41, 89], and forbidden-poset type properties in matrices [69], are hereditary (as all of them can be characterized by a finite set of forbidden submatrices), so Theorem 2.7 gives a new proof that these properties, and many of their natural extensions, are constant-query testable. Naturally, our general tests are much less efficient than the tests specifically tailored for each of these properties (in terms of dependence of the underlying constants on the parameters of the problem), but the advantage of our result is its generality, that is, the fact that it applies to any hereditary property. Thus, for example, for any fixed ordered graph H and any integer k, the property that an ordered graph G admits a k-edge coloring with no monochromatic (ordered) induced copy of H is constant-query testable. As mentioned above, Ramsey properties of this type have been considered in the combinatorics literature, see [57] and the references therein. Another family of examples includes properties of (integer) intervals on the line. Any interval can be encoded by an edge connecting its two endpoints, where the order on the vertices (the endpoints) is the usual order on the real line. A specific example of a hereditary property is that the given set of intervals is closed under intersection. The forbidden structure is a set of 4 vertices $i < j < k < l$ where ik and jl are edges (representing intervals) whereas jk is a non-edge.

Finally, there are various examples of unordered hereditary graph properties that have simple representations using a small *finite* forbidden family of *ordered* subgraphs, while in the unordered representation, the forbidden family is infinite. Some examples of such properties are bipartiteness, being a chordal graph, and being a strongly chordal graph [39, 59]. For such properties, when the input graph is supplied with the "right" ordering of the vertices, one can derive the strong testability using the version of Theorem 2.6 for *finite* families of forbidden ordered subgraphs – see Theorem 2.34 below – instead of using the infinite unordered version, Theorem 2.2.

Related and Subsequent results

Several additional results exploring the landscape of property testing in ordered structures have been established by the author and multiple coauthors. In [6], Alon and the author prove several removal lemma type results for ordered and partially ordered binary matrices, in which the dependence between the parameters in the removal lemma is *polynomial* (rather than tower-type or worse as is inherent when using Szemerédi regularity; see discussion below). The results there rely on the efficient regularity lemma for binary matrices [8], and extend the (unordered) testability results from the same paper. In [24], the author and Fischer study regularity-based transformations of constant-query tests to constant-query tolerant tests in ordered structures, thereby generalizing well-known results in the unordered context [9, 70]. In [25], the author, Fischer, Levi and Yoshida develop a limit object for ordered graphs and matrices, the *orderon*, also discussing implications for property testing. One such implication is an analytical proof of the ordered graph (and matrix) removal lemma presented here.

Discussion and open questions

Several possible directions for future research follow from the results.

Dependence between the parameters of the ordered removal lemmas Our proofs rely heavily on strong variants of the graph regularity lemma. Regularity-based proofs generally have a notoriously bad dependence between the parameters of the problem. In the notation of Theorem 2.6, for a fixed finite family $\mathcal{F}$ of forbidden ordered subgraphs, δ^{-1} is generally very large in terms of ε^{-1}, meaning that the number of queries required for the corresponding test for such properties is very large in terms of ε^{-1}. Indeed, the original Szemerédi regularity lemma imposes a tower-type dependence between these parameters [75, 88, 104], while the variant we use is at least as strong (and at least as expensive) as the strong regularity lemma [7], which is known to have a wowzer (tower of towers) type dependence between its parameters [55, 93]. Note that for infinite families $\mathcal{F}$ the dependence between the parameters may be arbitrarily bad [11].

In a breakthrough result of Fox [73], the first known proof for the (unordered) graph removal lemma that does not use the regularity lemma is given. However, the dependence between the parameters there is still of a tower type. In any case, it will be interesting to try to obtain a proof for the ordered case, that does not go through the strong regularity needed in our proof.

Better dependence for specific properties As discussed above, for ordered binary matrices there is an efficient conditional regularity lemma [8], in which the dependence of δ^{-1} on ε^{-1} is polynomial. It will be interesting to try to combine the ideas from our proof with this binary matrix regularity lemma, to obtain a removal lemma for finite families of ordered binary matrices with better dependence between the parameters. Ideally, one hopes for a removal lemma with polynomial dependence, but even obtaining such a lemma with, say, exponential dependence will be interesting. More generally, it will be interesting to find large families of hereditary ordered graph or matrix properties that have more efficient tests than those obtained from our results. See, e.g., [79] for recent results of this type for unordered graph properties.

Characterization of constant-query testable ordered properties For unordered graphs, Alon and Shapira [12] showed that a property is constant-query testable using an *oblivious* one-sided test, which is a test whose behavior is independent of the size of the input, *if and only if* the property is (almost) hereditary. It will be interesting to obtain similar characterizations in the ordered case. The work of the author and Fischer [24] provides such a characterization for two-sided error testability for *earthmover resilient* properties, which roughly speaking are all properties for which a slight change in the order of the base elements (i.e., vertices in a graph, or rows and columns in a matrix) does not substantially change the distance from the property. We believe that similarly to the unordered case, being testable with one-sided-error by a canonical test, that picks a random induced substructure and queries it, is roughly equivalent among earthmover resilient properties to being hereditary. Note that here, unlike the unordered setting, the restriction to canonical tests is essential; in the third part of this thesis we consider local properties that are not testable efficiently by a canonical test, but have very efficient locality-based tests.

Generalization to ordered hypergraphs and hypermatrices It will be interesting to obtain similar removal lemmas (and consequently, testing results) for the high-dimensional analogues of ordered graphs and matrices, namely ordered k-uniform hypergraphs and k-dimensional hypermatrices. Such results were proved for unordered hypergraphs [106, 118, 132].

2.2 Outline

A proof of a graph removal lemma typically goes along the following lines: First, the vertex set of the graph is partitioned into a "constant" (not depending on the input graph size itself) number of parts, and a corresponding *regularity scheme* is found. The regularity scheme essentially allows that instead of considering the original graph, one can consider a very simplified picture of a constant size structure approximately representing the graph. On one hand, the structure has to approximate the original graph in the sense that we can "clean" the graph, changing only a small fraction of the edges, so that the new graph will not contain anything not already "predicted" by the representing structure. On the other hand, the structure has to be "truthful", in the sense that everything predicted by it in fact already exists in the graph.

In the simplest case, just a regular partition given by the original Szemerédi Lemma would suffice. More complex cases, like [7, 12], require a more elaborate regularity scheme. In our case, Section 2.5 provides a regularity scheme that addresses both edge configuration and vertex order, combining a graph regularity scheme with a scheme for strings.

Given a regularity scheme, we have to provide the graph cleaning procedure, as well as prove that if the cleaned graph still contains a forbidden subgraph, then the original graph already contains a structure containing many such graphs (this will consist of some vertex sets referenced in the regularity scheme). In Section 2.5.3 we show how to use the scheme to prove the removal lemma and the testability theorem for the case of a finite family $\mathcal{F}$ of forbidden subgraphs, while in Section 2.6 we show how to extend it for the case of a possibly infinite family $\mathcal{F}$. The latter section also contains a formal definition of what it means for the regularity scheme to predict the existence of a forbidden subgraph, while for the finite case it is enough to keep it implicit.

To extract the regularity scheme we need two technical aids. One of which, in Section 2.4.2, is just a rounding lemma that allows us to properly use integer quantities to approximate real ones. While in many works the question of dealing with issues related to the divisibility of the number of vertices is just hand-waved away, the situation here is complex enough to merit a formal explanation of how rounding works.

In Section 2.4.1 we develop a Ramsey-type theorem that we believe to be interesting in its own right. The use of Ramsey-type theorems is prevalent in nearly all works dealing with regularity schemes, as a way to allow us to concentrate only on “well-behaved” structures in the scheme when we are about to clean the graph. Because of the extra complication of dealing with vertex-ordered graphs, we cannot just find Ramsey-type instances separately in different parts of the regularity scheme. Instead, we need to find the well-behaved structure “all at once”, and furthermore assure that we avoid enough of the “undesirable” parts where the regularity scheme does not reflect the graph. The fraction of undesirable features, while not large, must not depend on any parameters apart from the original distance parameter ε (and in particular must not depend on the size of the regularity scheme), which requires us to develop the new Ramsey-type theorem.

Roughly speaking, the theorem states the following: If we have a k-partite edge-colored graph with sufficiently many vertices in each part, then we can find a subgraph where the edges between every two parts are of a single color (determined by the identity of the two parts). However, we do it in a way that satisfies another requirement: If additionally the original graph is supplied with a set of “undesirable” edges comprising an α fraction of the total number of edges, then the subgraph we find will include not more than an $(1+\eta)\alpha$ fraction of the undesirable edges, for an η as small as we would like (in our application $\eta = 1$ will suffice).

2.2.1 Finding a Regularity Scheme

To prove the removal lemma we need a *regularity scheme*, that is a sequence of vertex sets whose “interaction” with the graph edges, and in our case also the graph vertex order, allows us to carry a cleaning procedure using combinatorial lemmas.

Historically, in the case of properties like triangle-freeness in ordinary graphs, a regular equipartition served well enough as a regularity scheme. One needs then to just remove all edges that are outside the reach of regularity, such as edges between the sets that do not form regular pairs. When moving on to more general properties of graphs, this is not enough. We need a robust partition (see [70]) instead of just a regular one, and then we can find a subset in each of the partition sets so that these “representative” sets will all form regular pairs. This allows us to decide what to do with problem pairs, e.g. whether they should become complete bipartite graphs or become edgeless (we also need to decide what happens *inside* each partition set, but we skip this issue in the sketch).

For vertex ordered graphs, a single robust partition will not do. The reason is that even if we find induced subgraphs using sets of this partition, there will be no guarantees about the vertex order in these subgraphs. The reason is that the sets of the robust partition could interact in complex ways with regards to the vertex order. Ideally we would like every pair of vertex sets to appear in one of the following two possible ways: Either one is completely before the other, or the two are completely “interwoven”.

To interact with the vertex order, we consider the robust partition along with a secondary interval partition. If we consider what happens between two intervals, then all vertices in one of the intervals will be before all vertices in the other one. This suggests that further dividing a robust partition according to intervals is a good idea. However, we also need that inside each interval, the relevant robust partition sets will be completely interwoven. In more explicit terms, we will consider what happens when we intersect them with intervals of a *refinement* of the original interval partition. If these intersections all have the "correct" sizes in relation of the original interval (i.e., a set that intersects an interval also intersects all relevant sub-intervals with sufficient vertex count), then we will have the "every possible order" guarantee.

Section 2.5 is dedicated to the formulation and existence proof of a regularity scheme suitable for ordered graphs. In Section 2.5.1 we present the concept of *approximating partitions*, showing several useful properties of them. Importantly, the notion of a robust partition is somewhat preserved when moving to a partition approximating it.

In Section 2.5.2 we develop the lemma that gives us the required scheme. Roughly speaking, it follows the following steps.

- We find a base partition P of the graph G, robust enough with regards to the graph edge colors, so as to ensure that it remains robust even after refining it to make it fit into a secondary interval partition.
- We consider an interval partition J of the vertex set V of G, that is robust with respect to P. That is, if we partition each interval of J into a number of smaller intervals (thus obtaining a refinement J'), most of the smaller intervals will contain about the same ratio of members of each set of P as their corresponding bigger intervals.
- Now we consider what happens if we construct a partition resulting from taking the intersections of the members of P with members of J'. In an ideal world, if a set of P intersects an interval of J, then it would intersect "nicely" also the intervals of J' that are contained in that interval. However, this is only mostly true. Also, this "partition by intersections" will usually not be an equipartition.
- We now modify a bit both P and J, to get Q and I that behave like the ideal picture, and are close enough to P and J. Essentially we move vertices around in P to make the intersections with the intervals in J' have about the same size inside each interval of J. We also modify the intersection set sizes (which also affects J a little) so they will all be near multiples of a common value (on the order of n). This is so we can divide them further into an equipartition that refines both the robust graph partition and the interval partition. The rounding Lemma 2.25 helps us here.

The above process generates the following scheme. Q is the modified base equipartition, and its size (i.e. number of parts) is denoted by k. I is the modified "bigger intervals" equipartition, and its size is denoted by m. We are allowed to require in advance that m will be large enough (that is, to have m bigger than a predetermined constant m_0). There is an equipartition Q' of size mt which refines both Q and I. That is, each part of Q' is fully contained in a part of Q and a part of I, and so each part of Q contains exactly mt/k parts of Q'. Moreover, there is the "smaller intervals" equipartition I' which refines I, and has size mb where $b = r(m, t)$ for a two-variable function r that we are allowed to choose in advance (r is eventually chosen according to the Ramsey-type arguments needed in the proof). Each part of I contains exactly b parts of I'. Finally, there is a "perfect" equipartition Q'' which refines Q' and I' and has size mbt, such that inside any bigger interval from I, the intersection of each part of Q' with each smaller interval from I' consists of exactly one part of Q''. Additionally, Q' can be taken to be very robust, where we are allowed to choose the robustness parameters in advance.

We are guaranteed that the numbers m and t are bounded in terms of the above function r, the robustness parameters, and m_0 for which we required that $m \geq m_0$. These bounds do not depend on the size of the input graph. See Lemma 2.33 for more details.

2.2.2 Proving a Finite Removal Lemma

Consider an ordered colored graph $G : \binom{[n]}{2} \to \Sigma$, and consider a regularity scheme consisting of equipartitions Q, I, Q', I', Q'' for G as described above.

We start by observing that if Q'' is robust enough, then there is a tuple W of "representatives" for Q'', satisfying the following conditions.

- For each part of Q'' there is exactly one representative, which is a subset of this part.
- Each representative is not too small: it is of order n (where the constants here may depend on all other parameters discussed above, but not on the input size n).
- All pairs of representatives are very regular (in the standard Szemerédi regularity sense).
- The densities of the colors from Σ between pairs of representatives are usually similar to the densities of those colors between the pairs of parts of Q'' containing them. Here the *density* of a color $\sigma \in \Sigma$ between vertex sets A and B is the fraction of σ-colored edges in $A \times B$.

Actually, the idea of using representatives, as presented above, was first developed in [7]. Note that each part of Q' contains exactly b representatives (since it contains b parts from Q'') and each small interval of I' contains exactly t representatives.

Now if Q' is robust enough then the above representatives for Q'' also represent Q' in the following sense: Densities of colors between pairs of representatives are usually similar to the densities of those colors between the pairs of parts of Q' containing them.

Consider a colored graph H whose vertices are the small intervals of I', where the "color" of the edge between two vertices (i.e. small intervals) is the $t \times t$ "density matrix" described as follows: For any pair of representatives, one from each small interval, there is an entry in the density matrix. This entry is the set of all colors from Σ that are dense enough between these two representatives, i.e., all colors whose density between these representatives is above some threshold.

An edge between two vertices of H is considered *undesirable* if the density matrix between these intervals differs significantly from a density matrix of the large intervals from I containing them. If Q' is robust enough, then most density matrices for pairs of small intervals are similar to the density matrices of the pairs of large intervals containing them. Therefore, the number of undesirables in H is small in this case.

Consider now H as an m-partite graph, where each part consists of all of the vertices (small intervals) of H that are contained in a certain large interval from I. We apply the undesirability-preserving Ramsey on H, and then a standard multicolored Ramsey within each part, to obtain an induced subgraph D of H with the following properties.

- D has exactly $d_{\mathcal{F}}$ vertices (small intervals) inside each part of H, where $d_{\mathcal{F}}$ is the maximum number of vertices in a graph from the forbidden family $\mathcal{F}$.
- For any pair of parts of H, all D-edges between these parts have the same "color", i.e. the same density matrix.
- For any part of H, all D-edges inside this part have the same "color".
- The fraction of undesirables among the edges of D is small.

Finally we wish to "clean" the original graph G as dictated by D. For any pair Q'_1, Q'_2 of (not necessarily distinct) parts from Q', let I_1, I_2 be the large intervals from I' containing them, and consider the density matrix that is common to all D-edges between I_1 and I_2. In this matrix there is an entry dedicated to the pair Q'_1, Q'_2, which we refer to as the set of colors from Σ that are "allowed" for this pair. The cleaning of G is done as follows: For every $u \in Q'_1$ and $v \in Q'_2$, if the original color of uv in G is allowed, then we do not recolor uv. Otherwise, we change the color of uv to one of the allowed colors.

It can be shown that if D does not contain many undesirables, then the cleaning does not change the colors of many edges in G. Therefore, if initially G is ε-far from $\mathcal{F}$-freeness, then there exists an induced copy of a graph $F \in \mathcal{F}$ in G with $l \le d_{\mathcal{F}}$ vertices after the cleaning. Considering our cleaning method, it can then be shown that there exist representatives $R_1, \ldots, R_l$ with the following properties. For any i, all vertices of R_i come before all vertices of R_{i+1} in the ordering of the vertices, and for any $i < j$, the color of $F(ij)$ has high density in $R_i \times R_j$. Recalling that all pairs of representatives are very regular, a well-known lemma implies that the representatives $R_1, \ldots, R_l$ span many copies of F, as desired.

From finite to infinite removal lemma

After the finite removal lemma is established, adapting the proof to the infinite case is surprisingly not difficult. The only problem of the finite proof is that we required D to have exactly $d_{\mathcal{F}}$ vertices in each large interval, where $d_{\mathcal{F}}$ is the maximal number of vertices of a graph in $\mathcal{F}$. This requirement does not make sense when $\mathcal{F}$ is infinite. Instead we show that there is a function $d_{\mathcal{F}}(m,t)$ that "plays the role" of $d_{\mathcal{F}}$ in the infinite case.

$d_{\mathcal{F}}(m,t)$ is roughly defined as follows: We consider the (finite) collection $\mathcal{C}(m,t)$ of all colored graphs with loops that have exactly m vertices, where the set of possible colors is the same as that of H (so the number of possible colors depends only on $|\Sigma|$ and t). We take $d_{\mathcal{F}}(m,t)$ to be the smallest number that guarantees the following. If a graph $C \in \mathcal{C}(m,t)$ exhibits (in some sense) a graph from $\mathcal{F}$, then C also exhibits a graph from $\mathcal{F}$ with no more than $d_{\mathcal{F}}(m,t)$ vertices.

The rest of the proof follows as in the finite case, replacing any occurrence of $d_{\mathcal{F}}$ in the proof with $d_{\mathcal{F}}(m,t)$. Here, if G contains a copy of a graph from $\mathcal{F}$ after the cleaning, then there is a set of no more than $d_{\mathcal{F}}(m,t)$ different representatives that are very regular in pairs and have the "right" densities with respect to some $F \in \mathcal{F}$ with at most $d_{\mathcal{F}}(m,t)$ vertices, so we are done as in the finite case.

From ordered graphs to ordered matrices To prove Theorem 2.8 for square matrices, we reduce the problem to a graph setting. Suppose that $M : U \times V \to \Sigma$ is a matrix, and add an additional color σ_0 to Σ. All edges between U and V will have the original colors from Σ, and edges inside U and inside V will have the new color σ_0. Note that we are not allowed to change colors to or from the color σ_0, as it actually signals "no edge". The proof now follows from the proof for graphs: We can ask the partition I into large intervals to "respect the middle", so all parts of I are either fully contained in U or in V. Moreover, colors of edges inside U or inside V are not modified during the cleaning step, and edges between U and V are not recolored to σ_0, since this color does not appear at all between the relevant representatives (and in particular, does not appear with high density).

To adapt the proof of Theorem 2.8 for non-square matrices, we need the divisibility condition to be slightly different than respecting the middle. In the case that $m = o(n)$, we need to construct two separate "large intervals" equipartitions, one for the rows and one for the columns, instead of one such equipartition I as in the graph case. The rest of the proof does not change.

2.3 Preliminaries and Definitions

In general, we may and will assume whenever needed throughout the chapter that n is large enough with respect to all relevant parameters. We generally denote "small" parameters and functions (whose values are always positive but can be arbitrarily close to zero) by small Greek letters[1], and "large" parameters and functions (whose values are always finite natural numbers but can be arbitrarily large) by Latin letters. We assume that all parameters in all statements of the lemmas are monotone in the "natural" direction, as in the following examples: $T(\alpha, b) \leq T(\alpha', b')$ for

[1]The only exception is λ, which will denote general real numbers, and ℓ which will denote their rounding.

$\alpha \geq \alpha', b \leq b'$, and $\gamma(c,\delta) \leq \gamma(c',\delta')$ for $c \geq c', \delta \leq \delta'$. We also assume that all "small" parameters are smaller than one, and all "large" parameters are larger than one.

We also assume that all functions are "bounded by their parameters", for example $\gamma(\alpha,k) \leq \alpha$ and $T(\alpha,k) \geq k$. These definitions extend naturally to any set of parameters, and are easily seen to be without loss of generality as long as we do not try to optimize bounds.

Colored graphs and charts A Σ-*colored graph* $G = (V, c_G)$ is defined by a totally ordered set of vertices V and a function $c_G : \binom{V}{2} \to \Sigma$. That is, G is a complete ordered graph whose edges are colored by elements of Σ. The standard notion of an (ordered) graph is equivalent to a $\{0,1\}$-colored graph. A Σ-*colored graph with loops* $G' = (V, c_{G'})$ is defined by a totally ordered set V and a function $c_{G'} : \binom{V}{2} \cup V \to \Sigma$. We identify the notation $c_{G'}(vv)$ with $c_{G'}(v)$ for any $v \in V$.

With a slight abuse of notation, we denote by $U_1 \times U_2 = \{\{u_1,u_2\} : u_1 \in U_1, u_2 \in U_2\}$ the set of edges between two disjoint vertex sets U_1 and U_2. A (k,Σ)-*chart* $C = (V_1,\dots,V_k,c_C)$ is defined by k disjoint vertex sets $V_1,\dots,V_k$ and a function $c_C : E_C \to \Sigma$, where $E_C = \bigcup_{1\leq i<j\leq k} U_i \times U_j$. In other words, it is an edge-colored complete k-partite graph. For C and G as above, we say that C is a *partition* of G if $V = \bigcup_{i=1}^{k} V_i$ and $c_G(e) = c_C(e)$ for any edge $e \in E_C$. Moreover, C is *equitable* if $||V_i| - |V_j|| \leq 1$ for any $1 \leq i,j \leq k$; an equitable partition is sometimes called an *equipartition.* The *size* $|C|$ of the partition C is the number of parts in it. For a partition C as above, a (k',Σ)-chart C' which is also a partition of G is said to be a G-*refinement* of C if we can write $C' = (V_{11},\dots,V_{1j_1},\dots,V_{k1},\dots,V_{kj_k},c_{C'})$ where $V_i = \bigcup_{l=1}^{j_i} V_{il}$. Note that $c_G(e) = c_C(e) = c_{C'}(e)$ for any edge $e \in E_C$. We sometimes omit the coloring from the description of a partition when it is clear from the context (as the coloring is determined by the partition of the vertices and the coloring of the graph). For two disjoint sets of vertices U, W and a coloring $c : U \times W \to \Sigma$, we say that the *density* of $\sigma \in \Sigma$ in (U,W,c) is $d_\sigma(U,W,c) = |(U \times W) \cap c^{-1}(\sigma)|/|U||W|$. the squared density is denoted by $d_\sigma^2(U,W,c)$. The *index* of (U,W,c) is

$$\mathrm{ind}(U,W,c) = \sum_{\sigma\in\Sigma} d_\sigma^2(U,W,c).$$

Note that $0 \leq \mathrm{ind}(U,V,c) \leq 1$ always holds. When the coloring c is clear from context, we usually simply write $d_\sigma(U,V)$ for density and $\mathrm{ind}(U,V)$ for index. For a chart C as above we define the *index* of C as

$$\mathrm{ind}(C) = \sum_{1\leq i<i'\leq k} \frac{|V_i||V_{i'}|}{\binom{|V|}{2}} \mathrm{ind}(V_i, V_{i'}, c\restriction_{V_i\times V_{i'}})$$

where $V = \bigcup_{i=1}^{k} V_i$. By the Cauchy-Schwarz inequality, for any two partitions C, C' of G where C' is a G-refinement of C we have

$$0 \leq \mathrm{ind}(C) \leq \mathrm{ind}(C') \leq 1. \tag{2.1}$$

For a function $f : \mathbb{N} \to \mathbb{N}$ and a constant $\gamma > 0$, we say that an equipartition C of size k is (f,γ)-*robust* if there exists no refining equipartition C' of C of size at most $f(k)$ for which

$\text{ind}(C') > \text{ind}(C) + \gamma$. The following observation states that for any colored graph G and any equipartition C of G, there exists an (f, γ)-robust equipartition C' refining C. The first explicit definition of robustness was given in [70].

Observation 2.10 (Robust partitioning of colored graphs [70])**.** *For any integer $k > 0$, function $f : \mathbb{N} \to \mathbb{N}$ and real $\gamma > 0$ there exists $T = T_{2.10}(k, f, \gamma)$ such that for any equipartition C of a colored graph G with $|C| = k$, there exists an (f, γ)-robust equipartition $C' = C'_{2.10}(C, f, \gamma)$ that refines C, where $|C'| \leq T$.*

Proof. Initially pick $C' = C$. Now, as long as C' is not (f, γ)-robust, let k' denote the number of parts of C'; we may replace C' by a G-refinement C'' of it with at most $f(k')$ parts and $\text{ind}(C'') > \text{ind}(C') + \gamma$. This process stops after at most $1/\gamma$ iterations, by inequality (2.1). □

The definition of robustness immediately implies the following.

Observation 2.11. *Let $P = (V_1, \dots, V_k)$ be an equipartition of a Σ-colored graph $G = (V, c)$, and suppose that P is $(f \circ g, \gamma)$-robust for two functions $f, g : \mathbb{N} \to \mathbb{N}$ and $\gamma > 0$. Then any equitable refinement of P with no more than $g(k)$ parts is (f, γ)-robust.*

The notion of robustness is stronger than the more commonly used notion of regularity. For a Σ-colored graph $G = (V, c)$ and an equipartition $P = (V_1, \dots, V_k)$ of G, a pair (V_i, V_j) is *ε-regular* if $|d_\sigma(V_i, V_j) - d_\sigma(V'_i, V'_j)| \leq \varepsilon$ for any $\sigma \in \Sigma$ and $V'_i \subseteq V_i, V'_j \subseteq V_j$ that satisfy $|V'_i| \geq \varepsilon|V_i|, |V'_j| \geq \varepsilon|V_j|$. P is an *ε-regular partition* if all but at most $\varepsilon\binom{k}{2}$ of the pairs (V_i, V_j) are ε-regular. The following lemma states that robust partitions are also regular; a lemma like it is implicit in the ideas of the original proof of [131]. The original was formulated only for non-colored graphs ($\Sigma = \{0, 1\}$), but the extension to colored graphs is not hard (and was also done in prior work).

Lemma 2.12 ([131], see also [70])**.** *For any $\varepsilon > 0$ there exist $f = f^{(\varepsilon)}_{2.12} : \mathbb{N} \to \mathbb{N}$ and $\delta = \delta_{2.12}(|\Sigma|, \varepsilon) > 0$ such that any (f, δ)-robust equipartition P of a Σ-colored graph G is also ε-regular.*

The next lemma was first formulated (with different notation and without the extension to general Σ) in [7], but in a sense the basic idea was already used in implicitly proving Lemma 2.12 in [131]. It will be useful for us later.

Lemma 2.13 ([7], see also [70])**.** *For any $\varepsilon > 0$ there exists $\delta = \delta_{2.13}(|\Sigma|, \varepsilon) > 0$, so that for every $f : \mathbb{N} \to \mathbb{N}$, any (f, δ)-robust equipartition $P = (V_1, \dots, V_k)$ of a Σ-colored graph G, and any equitable refinement $P' = (V_{11}, \dots, V_{1b}, \dots, V_{k1}, \dots, V_{kb})$ of P where $kb \leq f(k)$, choosing the indexes so that $V_i = \bigcup_{r=1}^{b} V_{ir}$ for any $i \in [k]$, it holds that*

$$\frac{1}{\binom{k}{2}b^2} \sum_{\sigma \in \Sigma} \sum_{i,i' \in [k]} \sum_{j,j' \in [b]} |d_\sigma(V_{ij}, V_{i'j'}) - d_\sigma(V_i, V_{i'})| \leq \varepsilon.$$

Another lemma that will be useful later is the following. This is Lemma 3.2 in [7].

Lemma 2.14 ([7])**.** *For any $\eta > 0$ there exists a function $\beta = \beta^{\eta}_{2.14} : \mathbb{N} \to \mathbb{N}$, so that for any integer $l > 0$ there exists $\kappa = \kappa_{2.14}(\eta, l)$ with the following property: If $H = ([l], c_H)$ and $V_1, \ldots, V_l$ are disjoint vertex sets of $G = (V, c)$, such that for any $i < j$, (V_i, V_j) is $\beta(l)$-regular and $d_{c_H(ij)}(V_i, V_j) \geq \eta$, then the number of induced H-copies in G with a vertex from V_i playing the role of vertex i of H is at least $\kappa \prod_{i=1}^{l} |V_i|$.*

Strings and intervals Consider an ordered set V whose elements are $v_1 < \ldots < v_n$. A *string* $S : V \to \Sigma$ is a mapping from the ordered set V to an alphabet Σ. An *interval partition* $I = (I_1, \ldots, I_k)$ of the string $S : V \to \Sigma$ is a partition $V = I_1 \ldots I_k$ into consecutive substrings of S: That is, there exist $0 = a_0 < \ldots < a_{k-1} < a_k = n$ such that $I_i = S(v_{a_{i-1}+1}) \ldots S(v_{a_i})$ for any $1 \leq i \leq k$. I is *equitable* (or an *interval equipartition*) if $a_i - a_{i-1} \in \{\lfloor n/k \rfloor, \lceil n/k \rceil\}$ for any $1 \leq i \leq k$. An *interval refinement* I' of I is an interval partition of S such that any part of I' is fully contained in a part of I. The *size* $|I|$ of an interval partition I is its number of parts.

Next we define notions of index and robustness that are suitable for strings and interval partitions. Similar notions were established in [17, 65]. For a string $S : V \to \Sigma$, the *density* of $\sigma \in \Sigma$ in S is $d_\sigma(S) = |S^{-1}(\sigma)|/|S|$ where $S^{-1}(\sigma) = \{v \in V : S(v) = \sigma\}$, and the squared density of σ in S is denoted by $d^2_\sigma(S)$. The *index* of S is $\text{ind}(S) = \sum_{\sigma \in \Sigma} d^2_\sigma(S)$. Finally, the *index* of an interval partition $I = (I_1, \ldots, I_k)$ of S is

$$\text{ind}(I) = \sum_{i=1}^{k} \frac{|I_i|}{|V|} \text{ind}(I_i).$$

As in the case of charts, for an interval equipartition I of a string S, we say that I is (f, γ)-robust if any interval equipartition I' of size at most $f(k)$ that refines I satisfies $\text{ind}(I') \leq \text{ind}(I) + \gamma$ (in the other direction, $\text{ind}(I) \leq \text{ind}(I')$ always holds). The following is an analogue of Observation 2.10, and its proof is essentially identical.

Observation 2.15 (Robust partitioning of intervals)**.** *For any integer $k > 0$, function $f : \mathbb{N} \to \mathbb{N}$ and real $\gamma > 0$ there exists $T = T_{2.15}(k, f, \gamma)$ such that any interval equipartition I of a string S where $|I| = k$ has an (f, γ)-robust interval refinement $I' = I'_{2.15}(I, f, \gamma)$ consisting of at most T intervals.*

The next lemma is an analogue of Lemma 2.13 for strings, and its proof is similar.

Lemma 2.16. *For any $\varepsilon > 0$ there exists $\delta = \delta_{2.16}(|\Sigma|, \varepsilon) > 0$, so that for every $f : \mathbb{N} \to \mathbb{N}$, any (f, δ)-robust interval equipartition $I = (I_1, \ldots, I_k)$ of a string $S : V \to \Sigma$, and any equitable interval refinement $I' = (I_{11}, \ldots, I_{1b}, \ldots, I_{k1}, \ldots, I_{kb})$ of I where $kb \leq f(k)$, choosing the indexes so that $I_i = \bigcup_{r=1}^{b} I_{ir}$ for any $i \in [k]$, it holds that*

$$\frac{1}{kb} \sum_{\sigma \in \Sigma} \sum_{i \in [k]} \sum_{j \in [b]} |d_\sigma(I_{ij}) - d_\sigma(I_i)| \leq \varepsilon.$$

We finish by defining the string that a partition of an ordered set induces on that set. The strings that we will consider in this chapter are of this type.

Definition 2.17 (String of a partition)**.** *For a partition $P = (V_1, \ldots, V_k)$ of an ordered set V, the P-*string *$S_P : V \to [k]$ maps any $v \in V$ to the element $i \in [k]$ such that $v \in V_i$.*

With slight abuse of notation, we will also use the notion of an interval partition in the context of ordered graphs; here each interval will simply be a set of consecutive vertices (with no accompanying function, in contrast to the case of strings).

2.4 Technical Aids

We develop here two tools that we will use for our proofs. The first tool is a Ramsey-type theorem that we believe to be interesting in its own right. We will use it to find a "uniform" structure with a global view on our graph. The second tool is a rounding lemma that allows us to evenly partition graphs also when the number of sets does not divide the number of vertices, without hand-waving away the divisibility issues (which might have been questionable in our context).

2.4.1 A Quantitative Ramsey-type Theorem

The multicolored Ramsey number $\mathrm{Ram}(s, k)$ is the minimum integer n so that in any coloring of K_n with s colors there is a monochromatic copy of K_k. It is well known that this number exists (i.e. is finite) for any s and k. For our purposes, we will also need a different Ramsey-type result, that keeps track of "undesirable" edges, as described in the following subsection.

Given a k-partite Σ-chart, we would like to pick a given number of vertices from each partition set, so that all edges between remaining vertices in each pair of sets are of the same color. However, in our situation we also have a "quantitative" requirement: A portion of the edges is marked as "undesirable", and we require that in the chart induced on the picked vertices the ratio of undesirable edges does not increase by much. Formally, we prove the following, which we state as a theorem because we believe it may have uses beyond the use in this thesis.

Theorem 2.18. *There exists a function $R_{2.18} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times (0, 1] \to \mathbb{N}$, so that if $G = (V_1, \ldots, V_k, c)$ is a k-partite Σ-chart with $n \geq R_{2.18}(|\Sigma|, k, t, \alpha)$ vertices in each class, and $B \subseteq \bigcup_{i<j \in [k]} (V_i \times V_j)$ is a set of "undesirable edges" of size at most $\varepsilon\binom{k}{2}n^2$, then G contains an induced subchart $H_{2.18}(G, B, t, \alpha) = (W_1, \ldots, W_k, c \restriction_{\bigcup_{1 \leq i < j \leq k}(W_i \times W_j)})$ with the following properties.*

- *$|W_i| = t$ for every $1 \leq i \leq k$.*
- *$c \restriction_{W_i \times W_j}$ is a constant function for every $1 \leq i < j \leq k$.*
- *The size of $B \cap (\bigcup_{1 \leq i < j \leq k}(W_i \times W_j))$ is at most $(1 + \alpha)\varepsilon\binom{k}{2}t^2$.*

In our use, these "vertices" would actually be themselves sets of a robust partition of the original graph, and "colors" will encode densities; an undesirable pair would have the "wrong" densities. Also, in our use case the undesirability of an edge will be determined solely by its color and the W_i that its end vertices belong to, which means that for each $1 \leq i < i' \leq k$ the edge set $W_i \times W_{i'}$ consist of either only desirable edges or only undesirable edges. When this happens, a later pick of smaller sets $W_i' \subset W_i$ will still preserve the ratio of undesirable edges (we will in fact perform such a pick using the original Ramsey's theorem inside each W_i). The following corollary summarizes our use of the theorem.

Definition 2.19. *Given a k-partite Σ-chart $G = (V_1, \ldots, V_k, c)$ and a set $B \subseteq \bigcup_{i<j\in[k]}(V_i \times V_j)$, we say that B is* orderly *if for every $1 \leq i < j \leq k$ there are no $e \in (V_i \times V_j) \cap B$ and $e' \in (V_i \times V_j) \setminus B$ for which $c(e) = c(e')$. In other words, the "position" and color of an edge fully determines whether it is in B.*

Corollary 2.20. *There exists a function $R_{2.20} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, so that if $G = (\bigcup_{i=1}^{k} V_i, c)$ is a Σ-colored graph with $|V_i| = n \geq R_{2.20}(|\Sigma|, k, t)$ for any $i \in [k]$ and $V_i \cap V_j = \emptyset$ for any $i \neq j \in [k]$, and $B \subseteq \bigcup_{i<j\in[k]}(V_i \times V_j)$ is an orderly set of "undesirable edges" of size at most $\varepsilon\binom{k}{2}n^2$, then G contains an induced subgraph D satisfying the following.*

- *The vertex set of D is $\bigcup_{i=1}^{l} U_i$ where $U_i \subseteq V_i$ and $|U_i| = t$ for any $i \in [k]$.*
- *For any $i \in [k]$, all edges inside U_i have the same color.*
- *For any $i < j \in [k]$, all edges in $U_i \times U_j$ have the same color.*
- *$\sum_{i<j\in[k]} |B \cap (U_i \times U_j)| \leq 2\varepsilon\binom{k}{2}t^2$.*

Proof. Take $R_{2.20}(s, k, t) = R_{2.18}(s, k, \mathrm{Ram}(s, t), 1)$ (recall that $\mathrm{Ram}(s, t)$ denotes the "traditional" s-colored Ramsey function). By Theorem 2.18, there exists a chart $H = (W_1, \ldots, W_k)$ with the following properties.

- $W_i \subseteq V_i$ and $|W_i| = \mathrm{Ram}(t, |\Sigma|)$ for every $i \in [k]$.
- For any pair $i < j \in [k]$, all edges in $W_i \times W_j$ have the same color.
- $\sum_{i<j\in[k]} |B \cap (W_i \times W_j)| \leq 2\varepsilon\binom{k}{2}\left(\mathrm{Ram}(t, |\Sigma|)\right)^2$.

Observe that for any pair $i < j \in [k]$, either $W_i \times W_j \subseteq B$ or $(W_i \times W_j) \cap B = \emptyset$, since all edges in $W_i \times W_j$ have the same color and B is orderly. Therefore, the number of pairs $i < j$ for which $(W_i \times W_j) \cap B \neq \emptyset$ is at most $2\varepsilon\binom{k}{2}$. Now we apply the traditional Ramsey's theorem inside each W_j to obtain a set $U_j \subseteq W_j$ of size t such that all edges inside W_j have the same color. Since $\sum_{i<j} |B \cap (U_i \times U_j)| \leq \sum_{i<j:(W_i\times W_j)\cap B\neq\emptyset} |B \cap (U_i \times U_j)| \leq 2\varepsilon\binom{k}{2}t^2$, the proof follows. □

Before moving to the proof of Theorem 2.18 itself, let us quickly note that a quantitative counterpart for the traditional (not k-partite) graph case does not exist (indeed, Corollary 2.20 is a way for us to circumvent such issues).

Proposition 2.21. *For any $\alpha > 0$, m, k, and large enough l, for infinitely many n there is a graph G and a set of undesirable pairs B, so that G has n vertices, B consists of at most $\frac{1}{mk}\binom{n}{2}$ pairs, G has no independent set of size l, and every clique of l vertices in G holds at least $(\frac{1}{m} - \alpha)\binom{l}{2}$ members of B.*

Proof. We construct G for any n that is a multiple of mk larger than lk. The graph G will be the union of k vertex-disjoint cliques, each with n/k vertices. In particular, G contains no independent set with l vertices, and any clique with l vertices must be fully contained in one of the cliques of G.

Now B will be fully contained in the edge-set of G, and will consist of the edge-set of mk vertex-disjoint cliques with n/mk vertices each, so that each of the cliques of G contains m of them. It is now not hard to see that any clique with l vertices in G will contain at least $(\frac{1}{m} - \alpha_l)\binom{l}{2})$ members of B, where $\lim_{l\to\infty} \alpha_l = 0$. □

Moving to the proof, the following is our main lemma. It essentially says that we can have a probability distribution over "Ramsey-configurations" in our chart that has some approximate uniformity properties.

Lemma 2.22. *There exists a function $R_{2.22} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times (0, 1] \to \mathbb{N}$, so that if $G = (V_1, \ldots, V_k, c)$ is a k-partite Σ-chart with $n \geq R_{2.22}(|\Sigma|, k, t, \delta)$ vertices in each class, then G contains a randomized induced subchart $H_{2.22}(G, t, \delta) = (W_1, \ldots, W_k, c \restriction_{\bigcup_{1\leq i<j\leq k}(W_i \times W_j)})$ satisfying the following properties.*

- *Either $|W_i| = t$ for every $1 \leq i \leq k$, or the chart is empty ($W_i = \emptyset$ for every i).*
- *$c \restriction_{W_i \times W_j}$ is a constant function for every $1 \leq i < j \leq k$ (with probability 1).*
- *For every $1 \leq i \leq k$, every $v \in V_i$ will be in W_i with probability at most t/n.*
- *For every $1 \leq i < j \leq k$, every $v \in V_i$ and every $w \in V_j$, the probability for both $v \in W_i$ and $w \in W_j$ to hold is bounded by $(t/n)^2$.*
- *The probability that the chart is empty is at most δ.*

Before we prove this lemma, we show how it implies Theorem 2.18.

Proof of Theorem 2.18. We set $R_{2.18}(a, k, t, \alpha) = R_{2.22}(a, k, t, \alpha/3)$. Given the k-partite Σ-chart G, we take the randomized subchart $H = H_{2.22}(G, t, \alpha/3) = (W_1, \ldots, W_k, c \restriction_{\bigcup_{1\leq i<j\leq k}(W_i \times W_j)})$, and prove that with positive probability it is the required subchart.

Let $B' = B \cap (\bigcup_{1\leq i<j\leq k} W_i \times W_j)$ denote the set of undesirable pairs that are contained in H. By the probability bound on pair containment and by the linearity of expectation, $\mathrm{E}[|B'|] \leq$

$(t/n)^2|B| \le \varepsilon\binom{k}{2}t^2$. Therefore, the probability for $|B'|$ to be larger than $(1+\alpha)\varepsilon\binom{k}{2}t^2$ is bounded by $\frac{1}{1+\alpha} \le 1-\alpha/2$. Therefore, with positive probability, both $|B'|$ is not too large and H is not the empty chart. Such an H is the desired subchart. $\square$

To prove Lemma 2.22 we shall make good use of the following near-trivial observation.

Observation 2.23. *There exists a function $m_{2.23} : \mathbb{N} \times \mathbb{N} \times (0,1] \to \mathbb{N}$, such that if A is a set of size at least $m_{2.23}(k,t,\delta)$ and $\mathcal{A} = (A_1,\dots,A_k)$ is a partition of A to k sets, then there exists a randomized subset $B = B_{2.23}(\mathcal{A},t,\delta)$ satisfying the following properties.*

- *Either $|B| = t$ or $B = \emptyset$.*
- *B is fully contained in a single A_i.*
- *For every $a \in A$, the probability for $a \in B$ is at most $t/|A|$.*
- *The probability for $B = \emptyset$ is at most δ.*

Proof. To choose the randomized subset B, first choose a random index I where $\Pr[I=i] = |A_i|/|A|$ for all $1 \le i \le k$. Next, if $|A_I| < t$ then set $B = \emptyset$, and otherwise set B to be a subset of size exactly t of A_I, chosen uniformly at random from all $\binom{|A_I|}{t}$ possibilities. Setting $m_{2.23}(k,t,\delta) = tk/\delta$, it is not hard to see that all properties for the random set B indeed hold. $\square$

Proof of Lemma 2.22. The proof is done by induction over k. The base case $k=1$ is easy – set $R_{2.22}(|\Sigma|,1,t,\delta) = t$, and let W_1 be a uniformly random subset of size t of V_1.

For the induction step from $k-1$ to k, we set $R_{2.22}(|\Sigma|,k,t,\delta) = m_{2.23}(|\Sigma|^s, r, \frac{1}{k+1}\delta)$, where $s = m_{2.23}(|\Sigma|^{k-1}, t, \frac{1}{k+1}\delta)$ and $r = R_{2.22}(|\Sigma|, k-1, t, \frac{1}{k+1}\delta)$. We set $W_1,\dots,W_k$ to be the result of the following random process.

First, we set $V'_1 \subseteq V_1$ to be a uniformly random subset of size exactly s. Then, for every $2 \le i \le k$, we set $V'_i \subseteq V_i$ to be the random set $B_{2.23}(\mathcal{V}_i, r, \frac{1}{k+1}\delta)$, where $\mathcal{V}_i$ is the partition of V_i obtained by classifying every $v \in V_i$ according to the colors $\langle c(w,v)\rangle_{w\in V'_1}$, i.e., two vertices in V_i are in the same partition set if all their pairs with vertices from V'_1 have the same colors.

If any of the V'_i came out empty, we set all W_i to $\emptyset$ and terminate the algorithm (this occurs with probability at most $\frac{k-1}{k+1}\delta$), and otherwise we continue. Note now, in particular, that for every $w \in V_1$ and $v \in V_i$ the probability for both $w \in V'_1$ and $v \in V'_i$ to hold is bounded by $(s/n)(r/n)$. This is since the probability guarantees of Observation 2.23 hold for any possible value of V'_1. Also, since each V'_i was independently drawn, for $v \in V_i$ and $w \in V_j$ (where $1 < i < j \le k$) the probability for both $v \in V'_i$ and $w \in V'_j$ to hold is bounded by $(r/n)^2$.

We now let H' denote the $(k-1)$-partite Σ-chart induced by $V'_2,\dots,V'_k$, and use the induction hypothesis to (randomly) set $W_2,\dots,W_k$ as the corresponding vertex sets of $H_{2.22}(H', t, \frac{1}{k+1}\delta)$. As before, if we receive empty sets then we also set $W_1 = \emptyset$ and terminate. Note that for $1 < i \le k$ and $v \in V'_i$, the probability for v to be in W_i is bounded by t/r. Hence, for $v \in V_i$, the probability for $v \in W_i$ to hold is bounded by $(r/n)(t/r) = t/n$. Similarly, for $1 < i < j \leq k$, for every $v \in V_i$ and $w \in V_j$ the probability for both $v \in W_i$ and $w \in W_j$ to hold is bounded by $(t/n)^2$. Also by similar considerations, for $v \in V_1$ and $w \in V_i$, the probability for both $v \in V_1'$ and $w \in W_i$ to hold is bounded by $(s/n)(t/n)$.

Finally, we set W_1 to be the random set $B_{2.23}(\mathcal{V}', t, \frac{1}{k+1}\delta)$, where $\mathcal{V}'$ is the partition of V_1' obtained by classifying each $v \in V_1'$ by the colors $\langle c(v, w)\rangle_{w \in W_i}$. Note that $c(v, w)$ in that expression depends only on v and the index i for which $w \in W_i$, because of how we chose each V_i' above. In particular, after the choice of W_1, the function $c \restriction_{W_1 \times W_i}$ is constant for each $1 < i \leq k$. Again, if we got an empty set for W_1, we set all $W_2, \ldots, W_k$ to be empty as well. By similar considerations as in the preceding steps, also here, for any $v \in V_1$ the probability of $v \in W_1$ is bounded by t/n, and for $w \in V_i$ where $1 < i \leq k$, the probability of both $v \in W_1$ and $w \in W_i$ is bounded by $(t/n)^2$.

The probability of obtaining empty sets in any of the steps is bounded by δ by a union bound, and all other properties of the random sets $W_1, \ldots, W_k$ have already been proven above. □

2.4.2 Multipartitions and Rounding

The following is a mechanism to handle "with one stroke" rounding issues throughout the chapter.

Definition 2.24 (Multipartitions). *A* multipartition *of a set L is a family M of subsets of L, that in particular includes L and all the singletons $\{i\}$ for $i \in L$, and furthermore every two sets $A, B \in M$ are either disjoint or one is contained in the other.*

To get an idea, an object that can be modeled as a multipartition is a partition of L (the multipartition would contain the partition sets, along with L itself and all singleton sets $\{i\}$), but also other objects, such as a partition and its refinement together, can be modeled as multipartitions. Here is the main lemma.

Lemma 2.25 (rounding feasibility). *If M and N are two multipartitions of the same set L, and $\lambda_i \in \mathbb{R}$ is a real value attached to every $i \in L$, then there exist integer values $\ell_i \in \mathbb{Z}$ attached to $i \in L$, satisfying the following.*

- $\ell_i \in \{\lfloor \lambda_i \rfloor, \lceil \lambda_i \rceil\}$ *for every $i \in L$.*
- $\sum_{i \in A} \ell_i \in \{\lfloor \sum_{i \in A} \lambda_i \rfloor, \lceil \sum_{i \in A} \lambda_i \rceil\}$ *for every $A \in M$ and for every $A \in N$.*
- $\sum_{i \in L} \ell_i \in \{\lfloor \sum_{i \in L} \lambda_i \rfloor, \lceil \sum_{i \in L} \lambda_i \rceil\}$.

Proof. Note that the middle item implies the other two (since M and N in particular include L and all singleton sets $\{i\}$ for $i \in L$). We define the following problem of solving a flow network with minimal and maximal constraints (for an exposition of flow networks see [136]).

- The node set of the flow network is $\{u_A : A \in M\} \cup \{w_A : A \in N\}$.

- The start node is u_L and the target node is w_L.
- For every $A, B \in M$, so that $A \subsetneq B$ and there is no $C \in M$ for which $A \subsetneq C \subsetneq B$, we put an edge from u_B to u_A with minimum flow $\lfloor \sum_{i \in A} \lambda_i \rfloor$ and maximum flow $\lceil \sum_{i \in A} \lambda_i \rceil$.
- For every $A, B \in N$, so that $A \subsetneq B$ and there is no $C \in N$ for which $A \subsetneq C \subsetneq B$, we put an edge from w_A to w_B with minimum flow $\lfloor \sum_{i \in A} \lambda_i \rfloor$ and maximum flow $\lceil \sum_{i \in A} \lambda_i \rceil$.
- For every $i \in L$ we put an edge from u_i to w_i with minimum flow $\lfloor \lambda_i \rfloor$ and maximum flow $\lceil \lambda_i \rceil$.
- We require the total flow of the network to be between $\lfloor \sum_{i \in L} \lambda_i \rfloor$ and $\lceil \sum_{i \in L} \lambda_i \rceil$.

This flow network has a real-valued solution by assigning λ_i flow to each edge of the type u_i, w_i, and then assigning the corresponding sums to all other network edges. Hence (since all constraints are integer-valued), the flow network has an integer-valued solution as well (see, e.g., the analysis of Lawler's algorithm in [136], page 602). Fixing such a solution, and setting ℓ_i to be the flow in the edge u_i, w_i for every $i \in L$, completes the proof. $\square$

An example of using the lemma is when we want to round the values in a 2-dimensional matrix so that the row sums and column sums are also rounded versions of the original sums (and in particular equal to the original sums if they happen to be integers). In our use the resulting integer values would be set sizes for an equipartition, that in turn refines other partitions with set size requirements.

We also note here that the statement of this lemma is false when we are presented with three multipartitions. Take for example the 3-dimensional matrix of size $2 \times 2 \times 2$, where $\lambda_{111} = \lambda_{100} = \lambda_{010} = \lambda_{001} = \frac{1}{2}$, with all other λ values being zero. Also for each of the three dimensions take the partition into two axes-parallel planes. The values on every set of every partition sum up to 1, and yet there are no corresponding $\ell_{ijk} \in \{\lfloor \lambda_{ijk} \rfloor, \lceil \lambda_{ijk} \rceil\}$ satisfying these constraints.

2.5 A Regularity Scheme for Ordered Graphs

2.5.1 The Approximating Partition Framework

Definition 2.26 (δ-approximating partitions). *Let $P = (V_1, \dots, V_k)$ and $Q = (U_1, \dots, U_l)$ be partitions of a set V of size n. We say that Q is a* δ-approximation *of P, or equivalently, that P and Q are* δ-close*, if there exists $T \subseteq V$ with $|T| \le \delta n$ such that $V_i \setminus T = U_i \setminus T$ for any $1 \le i \le \max\{k, l\}$, where for $i > k$ we define $V_i = \phi$, and similarly $U_i = \phi$ for $i > l$.*

Lemma 2.27. *For any $\varepsilon > 0$ there exists $\delta = \delta_{2.27}(\varepsilon) > 0$ such that any two δ-close partitions P and Q of (the vertex set of) a colored graph $G = (V, c)$ satisfy $|\mathrm{ind}(P) - \mathrm{ind}(Q)| \le \varepsilon$.*

Proof. Let $P = (V_1, \ldots, V_k)$ and $Q = (U_1, \ldots, U_l)$ be δ-close partitions of G, where we assume w.l.o.g. that $k < l$. For any $1 \le i \le k$ let $W_i = V_i \cap U_i$, and observe that $\sum_{i=1}^k |W_i| \ge (1-\delta)n$. we say that i is *bad* if $|W_i| \le (1-\sqrt{\delta})\min\{|V_i|, |U_i|\}$ and *good* otherwise. Then $\sum_{i \text{ bad}} |V_i| \le \sqrt{\delta}n$ and $\sum_{i \text{ bad}} |U_i| \le \sqrt{\delta}n$. When i and j are both good, we have

$$\big|\mathrm{ind}(V_i, V_j) - \mathrm{ind}(W_i, W_j)\big| \le \sum_{\sigma\in\Sigma} \big|(d_\sigma^2(V_i, V_j) - d_\sigma^2(W_i, W_j)\big| \le 2\sum_{\sigma\in\Sigma}\big|d_\sigma(V_i, V_j) - d_\sigma(W_i, W_j)\big|$$

$$\le 2\sum_{\sigma\in\Sigma}\left(d_\sigma(V_i, V_j)\left(\frac{|V_i||V_j|}{|W_i||W_j|} - 1\right) + \frac{|c^{-1}(\sigma) \cap ((V_i \times V_j) \setminus (W_i \times W_j))|}{|W_i||W_j|}\right) = O(\sqrt{\delta})$$

where the second inequality holds since $d_\sigma(V_i, V_j) + d_\sigma(U_i, U_j) \le 2$, the third inequality follows from the fact that $|x - y| \le z - x + z - y$ for $z \ge \max\{x, y\}$ and the last inequality follows from the observation that $|V_i||V_j| = (1 + O(\sqrt{\delta}))|W_i||W_j|$. Similarly, it holds that $|\mathrm{ind}(U_i, U_j) - \mathrm{ind}(W_i, W_j)| = O(\sqrt{\delta})$, so $|\mathrm{ind}(V_i, V_j) - \mathrm{ind}(U_i, U_j)| = O(\sqrt{\delta})$ when i, j are good. We finish by observing that

$$\mathrm{ind}(P) - \mathrm{ind}(Q) \le \sum_{i<j \text{ good}}\left(\frac{|V_i||V_j|}{\binom{n}{2}}\mathrm{ind}(V_i, V_j) - \frac{|U_i||U_j|}{\binom{n}{2}}\mathrm{ind}(U_i, U_j)\right) + 2\sqrt{\delta} = O(\sqrt{\delta})$$

where the first inequality holds since $\sum_{i \text{ bad}}\sum_{j\ne i}|V_i||V_j|ind(V_i, V_j)/\binom{n}{2} \le 2\sqrt{\delta}$ and the second inequality is true since $|V_i||V_j| = \left(1 + O(\sqrt{\delta})\right)|U_i||U_j|$ and $\mathrm{ind}(V_i, V_j) = \left(1 + O(\sqrt{\delta})\right)\mathrm{ind}(U_i, U_j)$ when i and j are good, and since $\mathrm{ind}(Q) \le 1$. Therefore, taking a suitable $\delta = \Theta(\varepsilon^2)$ in the statement of the lemma suffices. $\square$

Lemma 2.28. *Let P, Q be δ-close equipartitions of a colored graph G, where $|P| = |Q|$. Then any equitable refinement Q' of Q is δ-close to an equitable refinement P' of P, with $|P'| = |Q'|$.*

Proof. Write $P = (V_1, \ldots, V_k), Q = (U_1, \ldots, U_k), Q' = (U_{11}, \ldots, U_{1r}, \ldots, U_{k1}, \ldots, U_{kr})$ where $U_i = \bigcup_{j=1}^r U_{ij}$. Also, for any i, j let $W_i = V_i \cap U_i$ and $W_{ij} = V_i \cap U_{ij}$. Then $\sum_{i=1}^k\sum_{r=1}^r |W_{ij}| = \sum_{i=1}^k |W_i| \ge (1-\delta)n$, so we may take a refinement $P' = (V_{ij})_{1\le i\le k, 1\le j\le r}$ of P as follows: For any i, j we take V_{ij} that contains W_{ij} and ℓ_{ij} arbitrary additional elements from $V_i \setminus W_i$, where ℓ_{ij} is chosen by using Lemma 2.25 in the following manner.

For $1 \le i \le k$ and $1 \le j \le r$ we set $\lambda_{ij} = |V_i|/r - |W_{ij}|$. We set the multipartition M to consist of all singleton sets $\{ij\}$, the set $[k] \times [r]$, and the sets $\{i\} \times [r]$ for $1 \le i \le k$. We set the multipartition N to be the trivial one, just the singleton sets and $[k] \times [r]$. Invoking Lemma 2.25, we claim the following about the resulting ℓ_{ij}: Since $\sum_{j=1}^r \lambda_{ij} = |V_i| - |W_i|$, which is an integer, this will also equal the corresponding sum $\sum_{j=1}^r \ell_{ij} = |V_i| - |W_i|$, so we can get this way a refinement of P. Also, for any integer m (in our case $|V_i|$) it holds that $\lfloor \frac{m}{r} \rfloor = \lceil \frac{m+1}{r} \rceil - 1$, so the resulting V_{ij} would form an equipartition. The last issue that we need to deal with is when we have $\ell_{ij} = -1$ for some i and j, which could in fact happen. We claim however that in such a case we can move to another solution for which $\ell_{ij} = 0$. To see this, we note that $\ell_{ij} = -1$ only if $|V_i|/r$ is not an integer, $|W_{ij}| = |U_{ij}| = \lceil |V_i|/r \rceil$, and $\ell_{ij} = \lfloor \lambda_{ij} \rfloor$. But in this case one can see that there exists $j' \neq j$ so that $\ell_{ij'} = \lceil \lambda_{ij'} \rceil > 0$, and so we can increase ℓ_{ij} by 1 at the expense of $\ell_{ij'}$. $\square$

Lemma 2.29. *For any $\varepsilon > 0$ there exists $\delta = \delta_{2.29}(\varepsilon) > 0$ such that the following holds: If P and Q are δ-close equipartitions of a colored graph G, P is (f, δ)-robust and $|P| = |Q|$, then Q is (f, ε)-robust.*

Proof. Let P, Q be equipartitions as in the statement and pick $\delta = \delta_{2.27}(\varepsilon/3)$. Consider an equitable refinement Q' of Q of size at most $f(|Q|) = f(|P|)$. By Lemma 2.28 there exists some equitable refinement P' of P which δ-approximates Q' where $|P'| = |Q'| \leq f(|P|)$. The robustness of P implies that $\text{ind}(P') - \text{ind}(P) \leq \delta \leq \varepsilon/3$. By Lemma 2.27, $|\text{ind}(P) - \text{ind}(Q)| \leq \varepsilon/3$ and $|\text{ind}(P') - \text{ind}(Q')| \leq \varepsilon/3$. We conclude that $\text{ind}(Q') - \text{ind}(Q) \leq \varepsilon$. Thus, Q is (f, ε)-robust. $\square$

The definition of δ-close partitions works exactly the same for interval partitions. We observe that interval equipartitions and their densities are mostly determined by the number of parts.

Observation 2.30. *Any two interval equipartitions I and J of $[n]$ into m parts are $\frac{m^2}{n}$-close to each other. In particular, for any f, m and ε, if n is large enough as a function of m and ε, then for such I and J the densities satisfy $\frac{1}{m}\sum_{i=1}^{m}\sum_{\sigma\in\Sigma}|d_\sigma(I_i) - d_\sigma(J_i)| \leq \varepsilon$.*

Proof. If I and J are two interval equipartitions of $[n]$ with $|I| = |J| = m$, then we can set $T = \bigcup_{i=1}^{m-1}[i\lfloor \frac{n}{m} \rfloor + 1, i\lceil \frac{m}{m} \rceil]$. Clearly $|T| < m^2$ and $I_i \setminus T = J_i \setminus T$ for every $i \in [m]$. The second part of the observation then follows easily from the first part for n large enough. $\square$

2.5.2 The Core Lemmas

Definition 2.31 (Least Common Refinement)**.** *For two partitions $P = (V_1, \ldots, V_k)$ and $Q = (U_1, \ldots, U_l)$ of a colored graph G, the* least common refinement (LCR) *$P \sqcap Q$ of P and Q is the partition $(V_1 \cap U_1, \ldots, V_1 \cap U_l, \ldots, V_k \cap U_1, \ldots, V_k \cap U_l)$ (after removing empty sets from the list).*

Note that even if P and Q are equitable, $P \sqcap Q$ is not necessarily equitable.

The following lemma allows us to combine an "order-respecting" interval partition and a robust graph partition. The last statement in the formulation (about even n and m) is not needed for the rest of the proofs concerning ordered graphs, but we will refer to it when we discuss ordered matrices.

Lemma 2.32. *For any $\delta > 0$ and positive integers k, m and b, there exists $\gamma = \gamma_{2.32}(\delta, k) > 0$ such that the following holds: If P is an equipartition of an n-vertex colored graph G (for $n \geq N_{2.32}(\delta, k, m, b)$) with $|P| = k$, and J is an interval equipartition of S_P of size m which is (f, γ)-robust, where $f(m) \geq mb$, then there exist an interval equipartition $I = I_{2.32}(\delta, P, m, b)$ of size m, an interval equipartition $I' = I'_{2.32}(\delta, P, m, b)$ of size mb which refines I, an equipartition $Q = Q_{2.32}(\delta, P, m, b)$ of size k which δ-approximates P, and an equipartition $Q' = Q'_{2.32}(\delta, P, m, b)$ of* *size at most $T_{2.32}(\delta, k, m)$ which is a refinement of both I and Q, all satisfying that the LCR $Q'' = I' \sqcap Q'$ is an equipartition of size $|Q''| = b|Q'| = |Q'||I'|/|I|$ (i.e., each set of Q' intersects "nicely" all subintervals of the interval of I that contains it).*

Furthermore, if m and n are even, then I "respects the middle", that is $\sum_{i=1}^{m/2} |I_i| = \frac{n}{2}$.

Proof. We denote $P = (V_1, \dots, V_k)$, and set $\gamma_{2.32}(\delta, k) = \delta_{2.16}(k, \delta/20)$. The sets of the original interval equipartition J will be denoted by $J_1, \dots, J_m$.

We denote the eventual intervals of I by $(I_1, \dots, I_m)$, denote the eventual intervals of I' by $(I_{11}, \dots, I_{1b}, \dots, I_{m1}, \dots, I_{mb})$ where $I_i = \bigcup_{j=1}^{b} I_{ij}$ for any $i \in [m]$. The eventual sets of Q will be denoted by $(U_1, \dots, U_k)$, the sets of Q' by $(W_{11}, \dots, W_{1t}, \dots, W_{m1}, \dots, W_{mt})$ where $I_i = \bigcup_{s=1}^{t} W_{is}$, and the eventual sets of Q'' will be denoted as $W_{ijs} = W_{is} \cap I_{ij}$. We pick $t = k\lceil 20/\delta \rceil$, and correspondingly $T_{2.32}(\delta, k, m) = mt$.

Before choosing the partition intervals and sets themselves, we will choose sizes for the sets, and also choose sets of indexes $K_1, \dots, K_k$ describing the connection of Q to its refinement Q'. That is, eventually we will have $U_a = \bigcup_{(is) \in K_a} W_{is}$ for every $a \in [k]$. Finally defining the eventual $K_{ia} = \{s : (is) \in K_a\}$, $U_{ia} = I_i \cap U_a$ and $U_{ija} = I_{ij} \cap U_a$, we will also have $U_{ia} = \bigcup_{s \in K_{ia}} W_{is}$ and $U_{ija} = \bigcup_{s \in K_{ia}} W_{ijs}$.

We next determine the sizes $|K_{ia}|$, which will be found through our first use of Lemma 2.25 (and some further processing). We set the following parameters and multipartitions.

- $\lambda_{ia} = t|J_i \cap V_a|/|J_i|$.
- N contains the singleton sets $\{(ia)\}$, the set $[m] \times [k]$, and the set $\{i\} \times [k]$ for every $i \in [m]$. Note that in particular $\sum_{a=1}^{k} \lambda_{ia} = t$ is an integer, so we also have $\sum_{a=1}^{k} \ell_{ia} = t$.
- M contains the singleton sets $\{(ia)\}$, the set $[m] \times [k]$, and the set $[m] \times \{a\}$ for every $a \in [k]$. Note that since $|J_i| = \frac{n}{m} \pm 1 = (1 \pm \frac{m}{n})\frac{n}{m}$ and $|V_a| = (1 \pm \frac{k}{n})\frac{n}{k}$, we have $\sum_{i=1}^{m} \lambda_{ia} = (1 \pm 2\frac{m+k}{n})\frac{mt}{k}$, which means that for n large enough all the sums $\sum_{i=1}^{m} \ell_{ia}$ will equal $\frac{mt}{k} \pm 1$ (note that $\frac{mt}{k}$ is an integer), and moreover the number of $a \in [k]$ for which $\sum_{i=1}^{m} \ell_{ia} = \frac{mt}{k} + 1$ will equal the number of $a \in [k]$ for which this value is $\frac{mt}{k} - 1$.

After obtaining the values ℓ_{ia} through Lemma 2.25, we obtain ℓ'_{ia} from ℓ_{ia} through the following process: For all a for which $\sum_{i=1}^{m} \ell_{ia} = \frac{mt}{k}$, we set $\ell'_{ia} = \ell_{ia}$ for all $i \in [m]$. Otherwise we each time take an a for which $\sum_{i=1}^{m} \ell_{ia} = \frac{mt}{k} + 1$ and an a' for which $\sum_{i=1}^{m} \ell_{ia'} = \frac{mt}{k} - 1$. We choose i for which $\ell_{ia} > \ell_{ia'}$ set $\ell'_{ia} = \ell_{ia} - 1$, $\ell'_{ia'} = \ell_{ia'} + 1$, and for all other i' we set $\ell'_{i'a} = \ell_{i'a}$ and $\ell'_{i'a'} = \ell_{i'a'}$. The resulting ℓ'_{ia} satisfy $\sum_{a=1}^{k} \ell'_{ia} = t$, $\sum_{i=1}^{m} \ell'_{ia} = \frac{mt}{k}$, and $\ell'_{ia} = \lambda_{ia} \pm 2$.

Now we construct disjoint $K_1, \dots, K_k \subset [m] \times [t]$ so that for every $i \in [m]$ and every $a \in [k]$ we have $|K_{ia}| = \ell'_{ia}$. By the equations on the sums of ℓ'_{ia} above this is doable, and results in $|K_a| = \frac{mt}{k}$ for every $a \in [k]$.

Next, we determine the sizes of the sets I_{ij} of I' and W_{ijs} of Q'', through a second use of Lemma 2.25. We set the following parameters and multipartitions, for determining $\ell_{ijs} = |W_{ijs}|$.

- We plainly set $\lambda_{ijs} = \lambda = \frac{n}{mbt}$ for all $i \in [m]$, $j \in [b]$ and $s \in [t]$. Since all values are the same, the ℓ_{ijs} will have value differences bounded by 1, as befits the equipartition Q''.
- M consists of the singletons, the set $[m] \times [b] \times [t]$, and the following.
  - The set $\bigcup_{s \in K_{ia}}(\{i\} \times \{j\} \times \{s\})$ for every $i \in [m]$, $j \in [b]$ and $a \in [k]$. This will make U_{ija} have size between $\lfloor \frac{|K_{ia}|}{t}|I_{ij}| \rfloor$ and $\lceil \frac{|K_{ia}|}{t}|I_{ij}| \rceil$ (see about $|I_{ij}|$ below).
  - The set $\{i\} \times \{j\} \times [t]$ for each $i \in [m]$ and $j \in [b]$. Eventually we will have $|I_{ij}| = \sum_{s=1}^{t} \ell_{ijs} \in \{\lfloor \frac{n}{mb} \rfloor, \lceil \frac{n}{mb} \rceil\}$, so I' will be equitable.
  - The set $\{i\} \times [b] \times [t]$ for each $i \in [m]$. Eventually we will have $|I_i| = \sum_{s=1}^{t} \sum_{j=1}^{b} \ell_{ijs} \in \{\lfloor \frac{n}{m} \rfloor, \lceil \frac{n}{m} \rceil\}$, so I will be equitable.
  - If n and m are both even, we also add the sets $[1, m/2] \times [b] \times [t]$ and $[m/2+1, m] \times [b] \times [t]$ to M. Eventually we will have $\sum_{i=1}^{m/2} |I_i| = \sum_{i=m/2+1}^{m} |I_i| = n/2$.
- N consists of the singletons, the set $[m] \times [b] \times [t]$, and the following.
  - The set $\{i\} \times [b] \times \{s\}$ for every $i \in [m]$ and $s \in [t]$. This will ensure that the eventual Q' is equitable.
  - The set $\bigcup_{s \in K_{ia}}(\{i\} \times [b] \times \{s\})$ for every $i \in [m]$ and $a \in [k]$. This will make every U_{ia} have size between $\lfloor \frac{|K_{ia}|}{t}|I_i| \rfloor$ and $\lceil \frac{|K_{ia}|}{t}|I_i| \rceil$.
  - The set $\bigcup_{(is) \in K_a}(\{i\} \times [b] \times \{s\})$ for every $a \in [k]$. This will ensure that the eventual Q is equitable.

After obtaining the values ℓ_{ijs} for the respective set sizes $|W_{ijs}|$, we finally construct the partitions themselves. First we construct I as the only interval partition for which $|I_i| = \sum_{s=1}^{t} \sum_{j=1}^{b} \ell_{ijs}$ for every $i \in [m]$, and I' as the only refinement of I for which $|I_{ij}| = \sum_{s=1}^{t} \ell_{ijs}$ for every $i \in [m]$ and $j \in [b]$. For every $i \in [m]$ and $s \in [t]$ let $b_{is} \in [k]$ be the index such that $(is) \in K_{b_{is}}$. We now go over the indexes $i \in [m]$ and $j \in [b]$, and partition the vertices of I_{ij} into the sets W_{ijs} so that as many members of $V_{b_{is}} \cap I_{ij}$ as possible will go into every W_{ijs}. When we can no longer assign vertices in this manner (because $|I_{ij} \cap V_b|$ will not necessarily equal $\sum_{b_{is}=b} \ell_{ijs}$), we assign the remaining vertices to complete the sets that do not yet have the correct size.

Having defined I, I' and Q'', we define Q' by setting $W_{is} = \bigcup_{j=1}^{b} W_{ijs}$ for every $i \in [m]$ and $s \in [t]$, and define Q by setting $U_a = \bigcup_{(is) \in K_a} W_{is}$. All properties of I, I', Q, Q' and Q'' immediately follow from the construction, apart from the relationship between Q and P that we still need to prove.

Because of the way we chose the sets W_{ijs} to maximize the number of vertices they contain from the "correct" sets of P, The partitions P and Q will be δ-close if

$$\sum_{i=1}^{m} \sum_{j=1}^{b} \sum_{a=1}^{k} \big||V_a \cap I_{ij}| - |U_a \cap I_{ij}|\big| \le \delta n.$$

Denote the densities according to the string S_P by $d_{P,a}(I_{ij})$ (where $a \in [k]$), and the densities according to S_Q by $d_{Q,a}(I_{ij})$. For n large enough, because I' is an equipartition (interval sizes differ by not more than 1), we have

$$\frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{b}\sum_{a=1}^{k}\big||V_a \cap I_{ij}| - |U_a \cap I_{ij}|\big| \le \frac{2}{mb}\sum_{i=1}^{m}\sum_{j=1}^{b}\sum_{a=1}^{k}|d_{P,a}(I_{ij}) - d_{Q,a}(I_{ij})|.$$

From now on we bound the sums on the right hand side. Recall that J denotes the original interval equipartition of size m, and let J' be any refinement of J of size mb. By Observation 2.30, for n large enough we have $\frac{1}{mb}\sum_{a\in[k]}\sum_{i\in[m]}\sum_{j\in[b]}|d_{P,a}(I_{ij}) - d_{P,a}(J_{ij})| \le \delta/20$. By Lemma 2.16, we know that $\frac{1}{mb}\sum_{a\in[k]}\sum_{i\in[m]}\sum_{j\in[b]}|d_{P,a}(J_{ij}) - d_{P,a}(J_i)| \le \delta/20$. Now, recall that we chose the sets K_a so that $|K_{ia}| = t \cdot d_{P,a}(J_i) \pm 2$. This means that $\frac{1}{m}\sum_{a\in[k]}\sum_{i\in[m]}|d_{P,a}(J_i) - d_{Q,a}(I_i)| \le \delta/5$ (recalling also how we chose t). Finally, by our construction, for n large enough, $\frac{1}{mb}\sum_{a\in[k]}\sum_{i\in[m]}\sum_{j\in[b]}|d_{Q,a}(I_i) - d_{Q,a}(I_{ij})| \le \delta/20$. This follows from the size restriction that we ensured for the sets U_{ia} and U_{ija}. Using triangle inequalities with all these bounds on the density differences concludes the proof. □

Lemma 2.33. *For any positive integer k, real value γ, functions $r : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{N}$, and any n-vertex ordered colored graph G (for large enough n), there exist an interval equipartition I into m parts where $k \le m \le S_{2.33}(\gamma, k, f, r)$, an equipartition Q' of G into mt parts (not necessarily an interval equipartition) which refines I and is additionally (f,γ)-robust, where $mt \le T_{2.33}(\gamma, k, f, r)$, and an interval equipartition I' into $m \cdot r(m,t)$ parts also refining I, so that the LCR $Q'' = Q' \sqcap I'$ is an equipartition into exactly $mt \cdot r(m,t)$ parts (so each set of Q' intersects "nicely" all relevant intervals in I').*

Moreover, if n is even, then m will be even and I will respect the middle.

Proof. For each $l \in \mathbb{N}$ we define a function $g_l : \mathbb{N} \to \mathbb{N}$ by setting for every $m \in \mathbb{N}$

$$g_l(m) = m \cdot r(m, T_{2.32}(\delta_{2.29}(\gamma), l, m)/m).$$

Then we define a function $h : \mathbb{N} \to \mathbb{N}$ setting for every $l \in \mathbb{N}$

$$h(l) = f(T_{2.32}(\delta_{2.29}(\gamma), l, T_{2.15}(k, g_l, \gamma_{2.32}(\delta_{2.29}(\gamma), l)))).$$

We start with an equipartition P that is $(h, \delta_{2.29}(\gamma))$-robust that we obtain by Observation 2.10, and then with respect to the string S_P we obtain by Observation 2.15 an interval equipartition J that has at least k parts and is $(g_{|P|}, \gamma_{2.32}(\delta_{2.29}(\gamma), |P|))$-robust. Note that $|P| \le T_{2.10}(1, h, \delta_{2.29}(\gamma))$, and hence $|J| \le T_{2.15}(k, g_{T_{2.10}(1,h,\delta_{2.29}(\gamma))}, \gamma_{2.32}(\delta_{2.29}(\gamma), T_{2.10}(1, h, \delta_{2.29}(\gamma))))$, which we set as our $S_{2.33}(\gamma, k, f, r)$.

If n is even, then we make sure that k is also even (otherwise we replace it with $k + 1$ in all of the above), and then $|J|$ will be even as well (and our subsequent use of Lemma 2.32 will provide an I that respects the middle).

We then invoke Lemma 2.32 to get our partitions $I = I_{2.32}(\delta_{2.29}(\gamma), P, |J|, g_{|P|}(|J|)/|J|)$, $I' = I'_{2.32}(\delta_{2.29}(\gamma), P, |J|, g_{|P|}(|J|)/|J|)$, and $Q' = Q'_{2.32}(\delta_{2.29}(\gamma), P, |J|, g_{|P|}(|J|)/|J|)$. By the size guarantees of Lemma 2.32 we have $|I| = |J|$ (ensuring our size bound for $|I|$), and $|Q'|$ is bounded by $T_{2.32}(\delta_{2.29}(\gamma), T_{2.10}(1, h, \delta_{2.29}(\gamma)), S_{2.33}(\gamma, k, f, r))$, which we set as our $T_{2.33}(\gamma, k, f, r)$.

Lemma 2.32 guarantees all requirements apart from the robustness of Q'. To prove it, we note that Q' is a refinement of the partition $Q = Q_{2.32}(\delta_{2.29}(\gamma), P, |J|, g_{|P|}(|J|)/|J|)$ into at most $T_{2.32}(\delta_{2.29}(\gamma), |P|, T_{2.15}(k, g_{|P|}, \gamma_{2.32}(\delta_{2.29}(\gamma), |P|)))$ parts, where $|Q| = |P|$ and Q and P are $\delta_{2.29}(\gamma)$-close. Hence by invoking Lemma 2.29 (which makes Q (h, γ)-robust), and then Observation 2.11, we get that Q' is indeed (f, γ)-robust. $\square$

2.5.3 The Finite Case for Graphs

This section contains the proof of Theorem 2.6 for the case that the forbidden family $\mathcal{F}$ is finite. This is the ordered generalization of the finite induced graph removal lemma (Theorem 2.1).

Theorem 2.34 (Finite ordered graph removal lemma)**.** *Fix a finite set Σ with $|\Sigma| \geq 2$. For any finite family $\mathcal{F}$ of ordered graphs $F : \binom{[n_F]}{2} \to \Sigma$ and any $\varepsilon > 0$ there exists $\delta = \delta(\mathcal{F}, \varepsilon) > 0$, such that any ordered graph $G : \binom{V}{2} \to \Sigma$ that is ε-far from $\mathcal{F}$-freeness contains at least δn^q induced copies of some graph $F \in \mathcal{F}$.*

The proof of Theorem 2.6 is completed in Section 2.6, by considering the case where $\mathcal{F}$ is infinite. The proof for the infinite case mostly relies on ideas and tools presented in this section, but requires another step, which is motivated by the ideas of Alon and Shapira [12] for the unordered case.

2.5.4 Representing Subsets

Fix a finite alphabet Σ and a finite family $\mathcal{F}$ over Σ. Let $d_{\mathcal{F}}$ denote the largest number of vertices in a graph from $\mathcal{F}$. Now let $G = (V, c)$ be an n-vertex Σ-colored graph and suppose that I, I', Q', Q'' are equipartitions of G of sizes m, mb, mt, mbt respectively, so that I, I' are interval partitions, I' and Q' refine I, and $Q'' = I' \sqcap Q'$. More specifically, we write $I = (I_1, \ldots, I_m)$, $I' = (I_{11}, \ldots, I_{1b}, \ldots, I_{m1}, \ldots, I_{mb})$, $Q' = (U_{11}, \ldots, U_{1t}, \ldots, U_{m1}, \ldots, U_{mt})$, $Q'' = (U_{111}, \ldots, U_{mbt})$, where $I_j = \bigcup_{r=1}^{b} I_{jr} = \bigcup_{s=1}^{t} U_{js}$ for any $j \in [m]$ and $U_{jrs} = I_{jr} \cap U_{js}$ for any $j \in [m], r \in [b], s \in [t]$. Note that this is the same setting as the one obtained in Lemma 2.33, but we do not apply the lemma at this point; in particular, we currently do not make any assumptions on the equipartitions other than those stated above. We may and will assume whenever needed that n is large enough (as a function of all relevant parameters), and that any tuple of subsets of V considered in this section has at least two parts (i.e., it is not trivial).

Definition 2.35 (Representing subsets)**.** *Let $\alpha, \beta, \mu > 0$ be real numbers and suppose that $A = (A_1, \ldots, A_l)$ is an equipartition of G. We say that $B = (B_1, \ldots, B_l)$* represents *A if $B_i \subseteq A_i$ for any $i \in [l]$. Furthermore, we say that B (α, β, μ)-*represents *A if the following holds.*

- *$B_i \subseteq A_i$ and $|B_i| \geq \alpha n$ for any $i \in [l]$.*

- *All pairs (B_i, B_j) with $i < j \in [l]$ are β-regular.*

- $\frac{1}{\binom{l}{2}} \sum_{i<j\in[l]} \sum_{\sigma\in\Sigma} |d_\sigma(B_i, B_j) - d_\sigma(A_i, A_j)| \leq \mu$.

The following lemma is a slight variant of Corollary 3.4 in [7], suggesting that partitions that are robust enough have good representing subsets. The proof follows along the same lines of the proof of Lemma 3.2 in [56], so we omit it.

Lemma 2.36 ([7, 56])**.** *For any $\mu > 0$ and function $\beta : \mathbb{N} \to (0,1)$ there exist a function $f = f_{2.36}^{(\beta,\mu)} : \mathbb{N} \to \mathbb{N}$ and a real number $\gamma = \gamma_{2.36}(\mu) > 0$, such that for any integer $l > 0$ there is a real number $\alpha = \alpha_{2.36}(\beta, \mu, l) > 0$, all satisfying the following. If $A = (A_1, \dots, A_l)$ is an (f, γ)-robust equipartition of G, then there exists a tuple $B = (B_1, \dots, B_l)$ which $(\alpha, \beta(l), \mu)$-represents A.*

The next lemma is not hard to derive from Lemma 2.36 using Lemma 2.13, and is more suitable to our setting.

Lemma 2.37. *For any function $\beta : \mathbb{N} \to (0,1)$, function $g : \mathbb{N} \to \mathbb{N}$, and real number $\mu > 0$, there exist a function $f = f_{2.37}^{\beta,g,\mu} : \mathbb{N} \to \mathbb{N}$ and a real number $\gamma = \gamma_{2.37}(\mu) > 0$, so that for any integer $l > 1$ there exists $\alpha = \alpha_{2.37}(\beta, g, \mu, l) > 0$ satisfying the following: If $A = (A_1, \dots, A_l)$ is an (f, γ)-robust equipartition of G and $A' = (A_{11}, \dots, A_{1L}, \dots, A_{l1}, \dots A_{lL})$ is an equitable refinement of A, where $lL \leq g(l)$ and $A_i = \bigcup_{j=1}^{L} A_{ij}$ for any $i \in [l]$, then there exists $B = (B_{11}, \dots, B_{1L}, \dots B_{l1}, \dots, B_{lL})$ which $(\alpha, \beta(lL), \mu)$-represents A', and satisfies*

$$\frac{1}{\binom{l}{2}L^2} \sum_{i<i'\in[l]} \sum_{j,j'\in[L]} \sum_{\sigma\in\Sigma} |d_\sigma(B_{ij}, B_{i'j'}) - d_\sigma(A_i, A_{i'})| \leq 2\mu.$$

Proof. Pick $f = f_{2.37}^{\beta,g,\mu} = f_{2.36}^{(\beta,\mu)} \circ g$ and $\gamma = \gamma_{2.37}(\mu) = \min\{\delta_{2.13}(|\Sigma|, \mu), \gamma_{2.36}(\mu)\}$. Also pick $\alpha = \alpha_{2.37}(\beta, g, \mu, l) = \alpha_{2.36}(\beta, \mu, g(l))$, and suppose that A is (f, γ)-robust. By Observation 2.11 and the fact that $|A'| = lL \leq g(l)$, we know that A' is $(f_{2.36}^{(\beta,\mu)}, \gamma_{2.36}(\mu))$-robust, so by Lemma 2.36 there exists a tuple $B = (B_{11}, \dots, B_{1L}, \dots B_{l1}, \dots, B_{lL})$ which $(\alpha_{2.36}(\beta, \mu, lL), \beta(lL), \mu)$-represents A', and by the monotonicity of α, B also $(\alpha, \beta(lL), \mu)$-represents A'. In particular,

$$\frac{1}{\binom{l}{2}L^2} \sum_{i<i'\in[l]} \sum_{j,j'\in[L]} \sum_{\sigma\in\Sigma} |d_\sigma(B_{ij}, B_{i'j'}) - d_\sigma(A'_{ij}, A'_{i'j'})| \leq \mu.$$

Now by Lemma 2.13, and since $|A'| \leq g(l) \leq f(l)$,

$$\frac{1}{\binom{l}{2}L^2} \sum_{i<i'\in[l]} \sum_{j,j'\in[L]} \sum_{\sigma\in\Sigma} |d_\sigma(A'_{ij}, A'_{i'j'}) - d_\sigma(A_i, A_{i'})| \leq \mu.$$

Combining the above two inequalities and using the triangle inequality concludes the proof. $\square$

2.5.5 The Graph of the Representatives and its Coloring

For the next step, let $\Gamma = \Gamma(\Sigma, t)$ denote the collection of all $t \times t$ matrices M of the following form: Each entry of M is a non-empty subset of the color set Σ (where a subset is allowed to appear in multiple entries of M), so $|\Gamma(\Sigma, t)| < 2^{|\Sigma|t^2}$.

Definition 2.38 (Threshold color matrices, threshold graphs, undesirability)**.** *Suppose that $W = (W_{111}, \dots, W_{mbt})$ represents Q'' and define $W_{jr} = (W_{jr1}, \dots, W_{jrt})$ and $X_j = (U_{j1}, \dots, U_{jt})$ for any $j \in [m]$ and $r \in [b]$. Let $0 < \eta < \rho < 1/|\Sigma|$ be real numbers.*

For two t-tuples $A = (A_1, \dots, A_t)$ and $B = (B_1, \dots, B_t)$ where $A_s, B_s \subseteq V$ for any $s \in [t]$, the η-threshold matrix *$M = M(A, B, \eta) \in \Gamma$ of the pair A, B is the $t \times t$ matrix whose (s, s') entry (for $(s, s') \in [t]^2$) is the set of colors $\sigma \in \Sigma$ that satisfy $d_\sigma(A_s, B_{s'}, c\restriction_{A_s, B_{s'}}) \geq \eta$. Note that this set cannot be empty since $\eta < 1/|\Sigma|$.*

The (η, W)-threshold graph *H^η_W is an (ordered) Γ-colored graph defined as follows: The vertices of H^η_W are all parts of I', and the color of the edge $I_{jr}I_{j'r'}$ is $M(W_{jr}, W_{j'r'}, \eta)$.*

The edge $I_{jr}I_{j'r'}$ of H^η_W is ρ-undesirable if $j' > j$ and at least ρt^2 of the pairs $(s, s') \in [t]^2$ satisfy $M(X_j, X_{j'}, \rho)[s, s'] \nsubseteq M(W_{jr}, W_{j'r'}, \eta)[s, s']$. Finally, H^η_W is ρ-undesirable if at least $\rho\binom{m}{2}b^2$ of the edges $I_{jr}I_{j'r'}$ in it are ρ-undesirable, and ρ-desirable otherwise.

In other words, an edge $I_{jr}I_{j'r'}$ is undesirable if there are many pairs of sets $W_{jrs}, W_{j'r's'}$ in W, for which the density of some original edge color in $W_{jrs} \times W_{j'r's'}$ is significantly smaller than its density in $U_{js} \times U_{j's'}$. H^η_W is undesirable if it contains many undesirable edges. Note that the set of ρ-undesirable edges in H^η_W is orderly: Whether an edge $I_{jr}, I_{j'r'}$ of H^η_W is undesirable or not depends only on its color $M(W_{jr}, W_{j'r'}, \eta)$ and on $M(X_j, X_{j'}, \rho)$.

The following lemma relates the robustness of our partitions to the desirability of the resulting threshold charts.

Lemma 2.39. *For any $0 < \rho < 1/|\Sigma|$ and functions $\beta : \mathbb{N} \to (0, 1/|\Sigma|)$ and $g : \mathbb{N} \to \mathbb{N}$, there exist a function $f = f^{\rho,\beta,g}_{2.39} : \mathbb{N} \to \mathbb{N}$ and positive real numbers $\mu = \mu_{2.39}(\rho) \leq \rho$, $\gamma = \gamma_{2.39}(\rho)$ and $\alpha = \alpha_{2.39}(\rho, \beta, g, m, t)$, such that if Q' is (f, γ)-robust and $|Q''| \leq g(|Q'|)$, then there is a tuple $W = (W_{111}, \dots, W_{mbt})$ which $(\alpha, \beta(mbt), \mu)$-represents Q'', and furthermore $H^{\rho/2}_W$ is ρ-desirable.*

Proof. Let $0 < \rho < 1/|\Sigma|$ and suppose that $H^{\rho/2}_W$ is ρ-undesirable, where W is any tuple that represents Q''. The definition of undesirability implies that

$$\frac{1}{\binom{m}{2}t^2b^2} \sum_{j<j'\in[m]} \sum_{s,s'\in[t]} \sum_{r,r'\in[b]} \sum_{\sigma\in\Sigma} |d_\sigma(W_{jrs}, W_{j'r's'}) - d_\sigma(U_{js}, U_{j's'})| \geq \frac{\rho\binom{m}{2}b^2\rho t^2\rho/2}{\binom{m}{2}t^2b^2} = \frac{\rho^3}{2}. \quad (2.2)$$

Indeed, if $M(X_j, X_{j'}, \rho)[s, s'] \nsubseteq M(W_{jr}, W_{j'r'}, \rho/2)[s, s']$ then there exists some $\sigma \in \Sigma$ for which $d_\sigma(U_{js}, U_{j's'}) \geq \rho$ but $d_\sigma(W_{jrs}, U_{j'r's'}) \leq \rho/2$, so each such event contributes $\rho/2$ to the sum in the left hand side.

Therefore, $H_W^{\rho/2}$ is ρ-desirable if the above sum is smaller than $\rho^3/2$. Thus, we pick $\mu(\rho) = \rho^3/5$. Also pick $f_{2.39}^{\rho,\beta,g} = f_{2.37}^{\beta,g,\mu}$, $\gamma_{2.39}(\rho) = \gamma_{2.37}(\mu)$, and $\alpha_{2.39}(\rho, \beta, g, m, t) = \alpha_{2.37}(\beta, g, \mu, mt)$. Since Q' is $\left(f_{2.37}^{\beta,g,\mu}, \gamma_{2.37}(\mu)\right)$-robust, and since $|Q''| \leq g(|Q'|)$, Lemma 2.37 implies that there exists $W = (W_{111}, \dots, W_{mbt})$ which $(\alpha, \beta(mbt), \mu)$-represents Q', also guaranteeing that the left hand side of (2.2) is at most $2\mu < \rho^3/2$, so $H_W^{\rho/2}$ is ρ-desirable. □

Definition 2.40 (Nicely colored subgraph). *Let $W = (W_{111}, \dots, W_{mbt})$ be a tuple of subsets that represents Q'' and let $\eta > 0$. A subgraph $D = (\bigcup_{j=1}^m D_j, c_D)$ of H_W^η is said to be* nicely colored *if the following conditions hold.*

- *For any $j \in [m]$, $D_j \subseteq I_j$ and $|D_j| = d_{\mathcal{F}}$.*
- *For any fixed $j \in [m]$, all edges inside D_j have the same color from Γ, denoted by $C_{jj}^{(D)}$.*
- *For any fixed $j < j' \in [m]$, all edges between D_j and $D_{j'}$ have the same color from Γ, denoted by $C_{jj'}^{(D)}$.*

The next lemma follows directly from Corollary 2.20.

Lemma 2.41. *For any two integers $m, t > 0$ there exists $R = R_{2.41}(m, t)$ satisfying the following: If $b \geq R_{2.41}(m, t)$, then for any tuple $W = (W_{111}, \dots, W_{mbt})$ that represents Q'' and any $\eta > 0$ there exists a nicely colored subgraph $D = D_{2.41}(W, \eta)$ of H_W^η. Moreover, if H_W^η is ρ-desirable for some $\eta < \rho < 1/|\Sigma|$, then the number of ρ-undesirable edges in D is at most $2\rho\binom{m}{2}(d_{\mathcal{F}})^2$.*

Proof. Take $R_{2.41}(m, t) = R_{2.20}(2^{|\Sigma|t^2}, m, d_{\mathcal{F}}) > R_{2.20}(|\Gamma_t|, m, d_{\mathcal{F}})$. Since the set of ρ-undesirable edges in H_W^η is orderly, we may apply Corollary 2.20 on H_W^η, to get a nicely colored subgraph D of it. If H_W^η is ρ-desirable for some $\eta < \rho < 1/|\Sigma|$, then by definition it has at most $\rho\binom{m}{2}b^2$ ρ-undesirable edges, and so the last condition in Corollary 2.20 implies that D has at most $2\rho\binom{m}{2}(d_{\mathcal{F}})^2$ ρ-undesirable edges. □

2.5.6 Cleaning the Original Graph

Definition 2.42 (Cleaned graph). *Let $W = (W_{111}, \dots, W_{mbt})$ be a tuple of subsets which represents Q'', let $\eta > 0$, and suppose that D is a nicely colored subgraph of H_W^η. The* cleaned graph *$G' = G'(G, D) = (V, c')$ is defined as follows. For any $u < v \in V$ where $u \in I_{js}$ and $v \in I_{j's'}$, we set $c'(uv) = c(uv)$ if $c(uv) \in C_{jj'}^{(D)}[s, s']$, and otherwise we set $c'(uv)$ to an arbitrary color from $C_{jj'}^{(D)}[s, s']$.*

The next lemma states that if D comes from a desirable H_W^η, then $G'(G, D)$ is close to G.

Lemma 2.43. *Suppose that D is a nicely colored subgraph of some H_W^η with W representing Q'' and $0 < \eta < \rho$, such that at most $2\rho\binom{m}{2}d_{\mathcal{F}}^2$ edges of D are ρ-undesirable. Then $G' = G'(G, D)$ is $(7|\Sigma|\rho + 2/m)$-close to G, where $m = |I|$.*

Proof. Write $G' = (V, c')$ and let $\mathcal{J}$ denote the set of pairs $j < j' \in [m]$ such that $D_j \times D_{j'}$ contains an undesirable edge. An edge $e \in \binom{V}{2}$ may satisfy $c'(e) \neq c(e)$ only if at least one of the following holds (some of the inequalities stated below rely on the assumption that n is large enough).

1. e lies inside some part I_j of I. The number of such edges is $\sum_{j=1}^{m} \binom{|I_j|}{2} \leq m\binom{\lceil n/m \rceil}{2} < \frac{2}{m}\binom{n}{2}$.

2. $e \in I_{j_1} \times I_{j_2}$ where $(j_1, j_2) \in \mathcal{J}$. But $|\mathcal{J}| \leq 2\rho\binom{m}{2}$: The number of ρ-undesirable edges in D is exactly $|\mathcal{J}|d_{\mathcal{F}}^2$, since D is orderly (with respect to the parts $D_1, \ldots, D_m$) and has $d_{\mathcal{F}}$ vertices in each D_i. Thus, $|\mathcal{J}|d_{\mathcal{F}}^2 \leq 2\rho\binom{m}{2}d_{\mathcal{F}}^2$, which implies the desired inequality. Therefore, the number of edges e of of this type is less than $3\rho\binom{n}{2}$.

3. $e \in U_{js} \times U_{j's'}$ where $j < j' \in [m]$, $(j, j') \notin \mathcal{J}$, and $M(X_j, X_{j'}, \rho)[s, s'] \nsubseteq C_{jj'}(D)[s, s']$. But since the number of pairs $(s, s') \in [t]^2$ that satisfy this condition for a fixed $(j, j') \notin \mathcal{J}$ is at most ρt^2, only at most $3\rho|I_j||I_{j'}|/2$ of the edges $e \in I_j \times I_{j'}$ belong here, implying that the total number of edges of this type is less than $2\rho\binom{n}{2}$.

4. $e \in U_{js} \times U_{j's'}$ where $j < j' \in [m]$, $(j, j') \notin \mathcal{J}$, and $M(X_j, X_{j'}, \rho)[s, s'] \subseteq C_{jj'}(D)[s, s']$, but $d_{c(e)}(U_{js}, U_{j's'}) < \rho$. The number of such edges in $U_{js} \times U_{j's'}$ is at most $|\Sigma| \cdot \rho|U_{js}||U_{j's'}|$, and the total number of such edges is less than $2\rho|\Sigma|\binom{n}{2}$.

Therefore, the total number of edges e with $c(e) \neq c'(e)$ is less than $(7\rho|\Sigma| + 2/m)\binom{n}{2}$. □

Lemma 2.44. *Let $W = (W_{111}, \ldots, W_{mbt})$ be a tuple that represents Q'' and let $\eta > 0$. If D is a nicely colored subgraph of H_W^{η} and the cleaned $G'(G, D)$ contains a copy of some $F = ([n_F], c_F) \in \mathcal{F}$, then there exist $W_{j_1 r_1 s_1}, \ldots, W_{j_{n_F} r_{n_F} s_{n_F}} \in W$ with the following properties.*

- *For any $i \in [n_F - 1]$, either $j_{i+1} > j_i$, or $j_{i+1} = j_i$ and $r_{i+1} > r_i$.*

- *For any $i < i' \in [n_F]$ it holds that $d_{c_F(ii')}(W_{j_i r_i s_i}, W_{j_{i'} r_{i'} s_{i'}}) \geq \eta$.*

Proof. Suppose that $G'(G, D) = (V, c')$ contains a copy of F whose vertices in V are $v_1 < \ldots < v_{n_F}$. For any $i \in [n_F]$, let $j_i \in [m], s_i \in [t]$ be the indices for which $v_i \in I_{j_i s_i}$ and denote the vertices of D inside I_{j_i} by $D_{j_i} = \{I_{j_i r_{i1}}, \ldots, I_{j_i r_{i d_{\mathcal{F}}}}\}$, where $r_{i1} < \ldots < r_{i d_{\mathcal{F}}} \in [b]$ for any $i \in [n_F]$. Then for any $i, i' \in [n_F]$ and $l, l' \in [d_{\mathcal{F}}]$, for which either $i < i'$, or $i = i'$ and $l < l'$, it holds that $c_F(ii') = c'(v_i, v_{i'}) \in C_{j_i j_{i'}}^{(D)}[s_i, s_{i'}] = M(W_{j_i r_{il}}, W_{j_{i'} r_{i'l'}}, \eta)[s_i, s_{i'}]$, and so by definition $d_{c_F(ii')}(W_{j_i r_{il} s_i}, W_{j_{i'} r_{i'l'} s_{i'}}) \geq \eta$.

Therefore, the sets $W_{j_1 r_{11} s_1}, \ldots, W_{j_{n_F} r_{n_F n_F} s_{n_F}}$ satisfy the conditions of the lemma: They exist, since $n_F \leq d_{\mathcal{F}}$. The first condition holds since $j_1 \leq \ldots \leq j_{n_F}$, and if $j_i = j_{i+1}$ then $r_{ii} = r_{(i+1)i} < r_{(i+1)(i+1)}$. The second condition holds by the first paragraph of the proof (putting $l = i$ and $l' = i'$). □

2.5.7 Proof of Main Theorem

Suppose that G is ε-far from $\mathcal{F}$-freeness. Take the function $r = R_{2.41}$ (note that r is a two-variable function) and let $g : \mathbb{N} \to \mathbb{N}$ be defined by $g(l) = lr(l, l)$ for any $l \in \mathbb{N}$. Also take $k = \lceil 20/\varepsilon \rceil$, $\rho = \varepsilon/8|\Sigma|$, and $\beta : \mathbb{N} \to (0, 1/|\Sigma|)$ as a constant function that satisfies $\beta(l) = \beta^{\rho/2}_{2.14}(d_{\mathcal{F}})$ for any $l \in \mathbb{N}$. Also take $f = f^{\rho,\beta,g}_{2.39}$, and $\gamma = \gamma_{2.39}(\rho)$.

Apply Lemma 2.33 with parameters k, γ, r, f, obtaining the equipartitions I, I', Q', Q'' of sizes m, mb, mt, mbt as in the statement of the lemma, where $k \le m \le S_{2.33}(\gamma, k, f, r)$, $mt \le T_{2.33}(\gamma, k, f, r)$, $b = r(m, t) = R_{2.41}(m, t)$, and Q' is (f, γ)-robust. Observe that $|Q''| = mtr(m, t) \le g(mt) = g(|Q'|)$.

Next, define $\alpha = \alpha_{2.39}(\rho, \beta, g, S_{2.33}(\gamma, k, f, r), T_{2.33}(\gamma, k, f, r))$ and $\mu = \mu_{2.39}(\rho)$. By Lemma 2.39, and since $\beta(l) = \beta^{\rho/2}_{2.14}(d_{\mathcal{F}})$ for any $l \in \mathbb{N}$, there is a tuple W which $(\alpha, \beta^{\rho/2}_{2.14}(d_{\mathcal{F}}), \mu)$-represents Q'', and $H^{\rho/2}_W$ is ρ-desirable. By Lemma 2.41, and since $b = R_{2.41}(m, t)$, there is a nicely colored subgraph $D = D_{2.41}(W, \rho/2)$, containing at most $2\rho\binom{m}{2}(d_{\mathcal{F}})^2$ ρ-undesirable edges.

Lemma 2.43 implies that $G' = G'(G, D)$ is $(7|\Sigma|\rho + 2/m)$-close to G; but $7|\Sigma|\rho + 2/m \le 7\varepsilon/8 + 2/k < \varepsilon$, so G' contains a copy of some $F = ([n_F], c_F) \in \mathcal{F}$. Therefore, by Lemma 2.44 (putting $\eta = \rho/2$ in the statement of the lemma), there exist $W_{j_1 r_1 s_1}, \dots, W_{j_{n_F}, r_{n_F}, s_{n_F}} \in W$ that satisfy the conditions of the lemma. As all pairs of sets from W are $\beta^{\rho/2}_{2.14}(n_F)$-regular (since $n_F \le d_{\mathcal{F}}$), we can apply Lemma 2.14 to conclude that the number of F-copies in G is at least δn^q for $q = n_F \le d_{\mathcal{F}}$ and $\delta = \kappa_{2.14}(\rho/2, n_F)\alpha^{n_F} \ge \kappa_{2.14}(\rho/2, d_{\mathcal{F}})\alpha^{d_{\mathcal{F}}}$, concluding the proof.

2.6 The Infinite Case

In this section we use the same notation as in Section 2.5.3, unless stated otherwise. The proof of Theorem 2.6 follows that of Theorem 2.34 almost word by word, with only one major difference: In the proof of Theorem 2.34 we have picked $d_{\mathcal{F}}$ to be the largest number of vertices in a graph from $\mathcal{F}$, and showed that if G is ε-far from $\mathcal{F}$-freeness than there must be a set of at most $d_{\mathcal{F}}$ representatives of parts in Q'', that span a large number of F-copies for some $F \in \mathcal{F}$. However, in the infinite case, such a definition of $d_{\mathcal{F}}$ cannot work. Instead, we take $d_{\mathcal{F}}(m, t)$ to be a parameter that depends on the family $\mathcal{F}$, the size of the alphabet $|\Sigma|$ and the integers m, t (where $m = |I|$, $mt = |Q'|$). It is then shown that with this choice of $d_{\mathcal{F}}$, the proof follows similarly to the finite case, with Lemmas 2.41 and 2.44 being replaced with similar lemmas that are suitable for the infinite case (Lemmas 2.47 and 2.49 below, respectively).

2.6.1 Embeddability

Definition 2.45 (Embeddability)**.** *For a finite alphabet Σ, integers $m, t > 0$, $\Gamma(\Sigma, t)$-colored graph with loops $H = ([m], c_H)$ and Σ-colored graph $F = ([n_F], c_F)$, we say that F is* embeddable *in H if there exists a mapping $h : [n_F] \to V_H$ with the following properties.*

- *h is weakly order-preserving: $h(1) \le \ldots \le h(n_F)$.*

- *There exist integers $s_1, \ldots, s_{n_F} \in [t]$ so that $c_F(ii') \in c_H(h(i), h(i'))[s_i, s_{i'}]$ for any $i < i' \in [n_F]$.*

A family $\mathcal{F}$ of Σ-colored graphs is embeddable *in H if some $F \in \mathcal{F}$ is embeddable in H.*

The next lemma states that the desired $d_{\mathcal{F}}$ is indeed well-defined. It is similar in spirit to the ideas of Alon and Shapira [12] (see Section 4 there).

Lemma 2.46. *Fix a finite alphabet Σ. For any (finite or infinite) family $\mathcal{F}$ of Σ-ordered graphs and integers $m, t > 0$, there exists $d_{\mathcal{F}} = d_{\mathcal{F}}^{(2.46)}(m,t)$ with the following property. If $H = ([m], c_H)$ is a $\Gamma(\Sigma, t)$-colored graph with loops, and if $\mathcal{F}$ is embeddable in H, then there is a graph $F \in \mathcal{F}$ which is embeddable in H and has at most $d_{\mathcal{F}}^{(2.46)}(m,t)$ vertices.*

Proof. Let $\mathcal{H} = \mathcal{H}_{m,t}$ denote the set of all $\Gamma(\Sigma, t)$-colored graphs $H = ([m], c_H)$ with loops, such that $\mathcal{F}$ is embeddable in H. Note that $|\mathcal{H}_{m,t}| \le |\Gamma(\Sigma, t)|^{m^2} \le 2^{|\Sigma| t^2 m^2}$. For any $H \in \mathcal{H}$ let $\mathcal{F}_H \subseteq \mathcal{F}$ denote the collection of all graphs in $\mathcal{F}$ that are embeddable in H. Finally define

$$d_{\mathcal{F}}^{(2.46)}(m,t) = \max_{H \in \mathcal{H}_{m,t}} \min_{F \in \mathcal{F}_H} |F|$$

where $|F|$ denotes the number of vertices in F. Since $\mathcal{H}_{m,t}$ is finite, and since the set $\mathcal{F}_H$ is non-empty for any $H \in \mathcal{H}$ (by definition of $\mathcal{H}$), the function $d_{\mathcal{F}}^{(2.46)}(m,t)$ is well defined. Now let H be a graph as in the statement of the lemma and suppose that $\mathcal{F}$ is embeddable in H. Then $H \in \mathcal{H}_{m,t}$, so there exists $F \in \mathcal{F}_H$ of size at most $d_{\mathcal{F}}^{(2.46)}(m,t)$. $\square$

2.6.2 Adapting the Proof for Infinite Families

For what follows, a *nicely colored (m,t)-subgraph* is defined exactly like a nicely colored subgraph (see Definition 2.40), except that each set D_j is of size $d_{\mathcal{F}}^{(2.46)}(m,t)$. The following is a variant of Lemma 2.41 for the infinite case.

Lemma 2.47. *For any two integers $m, t > 0$ there exists $R = R_{2.47}(m,t)$ satisfying the following: If $b \ge R_{2.47}(m,t)$, then for any tuple $W = (W_{111}, \ldots, W_{mbt})$ that represents Q'' and any $\eta > 0$ there exists a nicely colored (m,t)-subgraph $D = D_{2.47}(W, \eta)$ of H_W^{η}. Moreover, if H_W^{η} is ρ-desirable for some $\eta < \rho < 1/|\Sigma|$, then the number of ρ-undesirable edges in D is at most $2\rho\binom{m}{2}(d_{\mathcal{F}}^{(2.46)}(m,t))^2$.*

The proof of Lemma 2.47 is essentially identical to that of Lemma 2.41, with any occurrence of $d_{\mathcal{F}}$ replaced by $d_{\mathcal{F}}^{(2.46)}(m,t)$. In particular we take $R_{2.47}(m,t) = R_{2.20}(2^{|\Sigma| t^2}, m, d_{\mathcal{F}}^{(2.46)}(m,t))$.

Next we state the variant of Lemma 2.43 for the infinite case. The proof is essentially identical.

Lemma 2.48. *Suppose that D is a nicely colored (m,t)-subgraph of some H_W^{η} with W representing Q'' and $0 < \eta < \rho$, such that at most $2\rho\binom{m}{2}(d_{\mathcal{F}}^{(2.46)}(m,t))^2$ edges of D are ρ-undesirable. Then $G' = G'(G, D)$ is $(7|\Sigma|\rho + 2/m)$-close to G, where $m = |I|$.*

The next lemma is the variant of Lemma 2.44 that we use in the infinite case. In contrast to the previous two lemmas, here the proof is slightly modified, and makes use of Lemma 2.46.

Lemma 2.49. *Let $W = (W_{111}, \dots, W_{mbt})$ be a tuple that represents Q'' and let $\eta > 0$. If D is a nicely colored (m,t)-subgraph of H^{η}_{W} and $G'(G,D)$ contains a copy of a graph from $\mathcal{F}$, then there exist $F = ([n_F], c_F) \in \mathcal{F}$, where $n_F \le d^{(2.46)}_{\mathcal{F}}(m,t)$, and sets $W_{j_1 r_1 s_1}, \dots, W_{j_{n_F} r_{n_F} s_{n_F}} \in W$, with the following properties.*

- *For any $i \in [n_F - 1]$, either $j_{i+1} > j_i$, or $j_{i+1} = j_i$ and $r_{i+1} > r_i$.*
- *For any $i < i' \in [n_F]$ it holds that $d_{c_F(ii')}(W_{j_i r_i s_i}, W_{j_{i'} r_{i'} s_{i'}}) \ge \eta$.*

Proof. Consider the Γ-colored graph with loops $D' = ([m], c_{D'})$: For any $j \le j'$, $c_{D'}(jj') = C^{(D)}_{jj'}$. Suppose that $G'(G,D) = (V, c')$ contains a copy of $A = ([n_A], c_A) \in \mathcal{F}$, whose vertices in V are $v_1 < \dots < v_{n_A}$. For any $i \in [n_A]$, let $j_i \in [m], s_i \in [t]$ be the indices for which $v_i \in I_{j_i s_i}$. Then for any $i < i' \in [n_A]$ we have $c_A(ii') = c'(v_i v'_i) \in C^{(D)}_{j_i j_{i'}}[s_i, s_{i'}] = c_{D'}(j_i j_{i'})[s_i, s_{i'}]$, and so A is embeddable in D' (by the mapping $i \mapsto j_i$). By Lemma 2.46, there exists $F = ([n_F], c_F) \in \mathcal{F}$ which is embeddable in D', where $n_F \le d^{(2.46)}_{\mathcal{F}}(m,t)$. Let $h : [n_F] \to D'$ denote a mapping that satisfies the conditions of Definition 2.45 and let $s'_1, \dots, s'_{n_F} \in [t]$ be the indices satisfying $c_F(ii') \in c_{D'}(h(i), h(i'))[s'_i, s'_{i'}]$ for any $i < i' \in [n_F]$.

For any $i \in [n_F]$ denote the vertices of D inside $I_{h(i)}$ by $I_{h(i) r'_{i1}}, \dots, I_{h(i) r'_{i d_{\mathcal{F}}(m,t)}}$, where $r'_{i1} < \dots < r'_{i d_{\mathcal{F}}(m,t)} \in [b]$ for any $i \in [n_F]$. The sets $W_{h(1) r'_{11} s'_1}, \dots, W_{h(n_F) r'_{n_F n_F} s'_{n_F}}$ satisfy the desired conditions: They exist, since $n_F \le d^{(2.46)}_{\mathcal{F}}(m,t)$, the first condition holds since h is order-preserving, and the second condition holds since $c_F(ii') \in c_{D'}(h(i), h(i'))[s'_i, s'_{i'}] = C^{(D)}_{h(i)h(i')}[s'_i, s'_{i'}]$. □

Proof of Theorem 2.6. The proof goes along the same lines as the proof of Theorem 2.34, but any occurrence of $d_{\mathcal{F}}$ in the proof of Theorem 2.34 and the accompanying lemmas is replaced here by $d^{(2.46)}_{\mathcal{F}}(m,t)$, including in the definitions of the functions β, r, and the term *nicely colored subgraph* is replaced by *nicely colored (m,t)-subgraph.* More specifically, here are the exact changes needed with respect to the proof of Theorem 2.34.

- We take the functions $\beta = \beta^{\rho/2}_{2.14}$ and $r = R_{2.47}$ (in the finite case we took β as a suitable constant function and $r = R_{2.41}$). The function g is defined as $g(l) = lr(l,l)$. Following the application of Lemma 2.33, we have $b = R_{2.47}(m,t)$.
- As in the the proof of Theorem 2.34, there is a tuple W which $(\alpha, \beta(mbt), \mu)$-represents Q'', and $H^{\rho/2}_{W}$ is ρ-desirable. By Lemma 2.47, and by our new choice of b, there is a nicely colored (m,t)-subgraph D of $H^{\rho/2}_{W}$, with at most $2\rho\binom{m}{2}\left(d^{(2.46)}_{\mathcal{F}}(m,t)\right)^2$ ρ-undesirable edges.
- Lemma 2.48 implies that G' contains a copy of a graph from $\mathcal{F}$. Now Lemma 2.49 implies the existence of sets $W_{j_1 r_1 s_1}, \dots, W_{j_{n_F}, r_{n_F}, s_{n_F}} \in W$ with $n_F \le d^{(2.46)}_{\mathcal{F}}(m,t)$, that satisfy the conditions of the lemma for $\eta = \rho/2$. Since all pairs of sets from W are $\beta^{\rho/2}_{2.14}(mbt)$-regular,

and since $mbt \geq b \geq d_{\mathcal{F}}^{(2.46)}(m,t) \geq n_F$, these pairs are also $\beta_{2.14}^{\rho/2}(n_F)$-regular. We apply Lemma 2.14 to get that the number of F-copies in G is at least δn^q for

$$\begin{aligned}
q &= n_F \leq d_{\mathcal{F}}^{(2.46)}(m,t) \leq d_{\mathcal{F}}^{(2.46)}(S_{2.33}(\gamma,k,f,r), T_{2.33}(\gamma,k,f,r)),\\
\delta &= \kappa_{2.14}\left(\rho/2, n_F\right)\alpha^{n_F} \geq \kappa_{2.14}\left(\rho/2, d_{\mathcal{F}}^{(2.46)}(m,t)\right)\alpha^{d_{\mathcal{F}}^{(2.46)}(m,t)}\\
&\geq \kappa_{2.14}(\rho/2, d_{\mathcal{F}}^{(2.46)}(S_{2.33}(\gamma,k,f,r), T_{2.33}(\gamma,k,f,r)))\alpha^{d_{\mathcal{F}}^{(2.46)}(S_{2.33}(\gamma,k,f,r),T_{2.33}(\gamma,k,f,r))}.
\end{aligned}$$

Indeed, the above bounds for q and δ depend only on $|\Sigma|, \varepsilon, \mathcal{F}$, and not on n. □

2.6.3 Adapting the Proof for Matrices

Finally we give a sketch of the proof of Theorem 2.8 for square matrices. The proof is very similar to the graph case, so we only describe why the proof for graphs also works here. Finally, we describe shortly how the proof can be adapted to the case of non-square matrices.

Proof sketch for Theorem 2.8. Given a square matrix $M : U \times V \to \Sigma$ where U, V are ordered, and a family $\mathcal{F}$ of forbidden submatrices, consider the Σ'-colored graph $G = (U \cup V, c)$ where $\Sigma' = \Sigma \cup \{\sigma_0\}$ for some $\sigma_0 \notin \Sigma$, and the union $U \cup V$ is ordered as follows: All elements of V come after all elements of U, and the internal orders of U and V remain as before. The edge colors of G satisfy $c(uv) = M(uv)$ for any $u \in U$ and $v \in V$, and $c(uv) = \sigma_0$ otherwise.

The proof now follows as in the graph case. It is important to note that while in the graph case one is allowed to change the color of any edge, here we are not allowed to change the color of an edge from or to the color σ_0. However, the proof still works, by the following observations. First, since $|U| = |V|$, the number of vertices in G is even, and so the interval partition I obtained here "respects the middle". That is, each part I_j of I will be fully contained in U or in V. Therefore, for every two parts $I_j, I_{j'}$ of I, either all edges in $I_j \times I_{j'}$ are colored by σ_0 or none of them is colored by σ_0. Second, it follows that the set of edges of the cleaned graph $G' = G'(G, D)$ that are colored by σ_0 is identical to that of G. In other words, to generate the cleaned graph we do not modify edge colors to or from σ_0. Since G is made $\mathcal{F}$-free only by modifying colors between U and V to other colors in Σ, one needs to modify at least $\varepsilon|U||V|$ edge colors, so the proof follows without changing the main arguments. □

The above proof works for square matrices, but it can be adapted to general $m \times n$ matrices: If $m = \Theta(n)$, then the condition on I needed is slightly different than respecting the middle, but this only slightly changes the structure of the equipartitions that we obtain via Lemma 2.33, without significantly affecting the proof. The proof can also be formulated for matrices with, say, $m = o(n)$ and $m = \omega(1)$, but then Lemma 2.33 needs to be especially adapted to accommodate the two "types" of vertices (row and column). Essentially we will have two interval equipartitions, one of the row vertices and one of the column vertices, along with their corresponding refinements.

Finally, the case where $m = \Theta(1)$ is essentially the case of testing one-dimensional strings; strings can be handled as per the discussion in Section 2.1.

It is important to note that one cannot use Theorem 2.6 as a black box to prove Theorem 2.8, as the distance of the graph G to $\mathcal{F}$-freeness might (potentially) be significantly smaller than ε, considering that the set of σ_0-colored edges in the $\mathcal{F}$-free graph that is closest to G might differ from the set of σ_0-colored edges in G.

Part II

Sublinear Algorithms for Sequential Pattern Detection

Chapter 3

Monotone Patterns: A Non-Adaptive $\Theta((\log n)^{\lfloor \log_2 k \rfloor})$ Algorithm

The results in this chapter appear in [22].

3.1 Introduction

For a fixed integer $k \in \mathbb{N}$ and a function (or sequence) $f\colon [n] \to \mathbb{R}$, a *length-$k$ monotone subsequence of f* is a tuple of k indices, $(i_1, \dots, i_k) \in [n]^k$, such that $i_1 < \cdots < i_k$ and $f(i_1) < \cdots < f(i_k)$. More generally, for a permutation $\pi\colon [k] \to [k]$, a *π-pattern of f* is given by a tuple of k indices $i_1 < \cdots < i_k$ such that $f(i_{j_1}) < f(i_{j_2})$ whenever $j_1, j_2 \in [k]$ satisfy $\pi(j_1) < \pi(j_2)$. A sequence f is π-free if there are no subsequences of f with order pattern π. Pattern avoidance and detection in an array is a central problem in theoretical computer science and combinatorics, dating back to the work of Knuth [97] (from a computer science perspective), and Simion and Schmidt [128] (from a combinatorics perspective); see also the survey [137]. Recently, Newman, Rabinovich, Rajendraprasad, and Sohler [109] (see [108] for the conference version) initiated the study of property testing for forbidden order patterns in a sequence. Their paper was the first to analyze algorithms for finding π-patterns in sublinear time (for various classes of the permutation π). The main motivation for testing order patterns arises in data-series analysis. In this context, a huge amount of continuous sequential data may arrive from various sources (e.g. sensors), with a need to develop algorithms that are as efficient as possible to understand the structural behavior of the data. Additional motivation naturally arises in combinatorics and other areas. See [109] for more details.

Of particular interest of π-freeness testing is the case where $\pi = (12\dots k)$, i.e., π is a monotone permutation. In this case, avoiding length-k monotone subsequence may be equivalently rephrased as being decomposable into $k-1$ monotone non-increasing subsequences, via Dilworth theorem [60]. Specifically, a function $f\colon [n] \to \mathbb{R}$ is $(12\dots k)$-free if and only if $[n]$ can be partitioned into $k-1$ disjoint sets $A_1, \dots, A_{k-1}$ such that, for each $i \in [k-1]$, the restriction $f|_{A_i}$ is non-increasing. When interested in algorithms for testing $(12\ldots k)$-freeness that have a *one-sided error*, the algorithmic task becomes the following. For $k \in \mathbb{N}$ and $\varepsilon > 0$, design a randomized algorithm that, given query access to a function $f\colon [n] \to \mathbb{R}$ guaranteed to be ε-far from being $(12\ldots k)$-free, outputs a length-k monotone subsequence of f with probability at least $2/3$.

The task above is a natural generalization of monotonicity testing of a function $f\colon [n] \to \mathbb{R}$ with algorithms that make a one-sided error, a question which dates back to the early works in property testing, and has received significant attention since in various settings (see, e.g., [1, 18, 32, 67, 82, 111, 138], Chapter 6 of this thesis, and the recent textbook [81]). For the problem of testing monotonicity, Ergün, Kannan, Kumar, Rubinfeld, and Viswanathan [63] were the first to give a non-adaptive algorithm which tests monotonicity of functions $f\colon [n] \to \mathbb{R}$ with one-sided error making $O(\log(n)/\varepsilon)$ queries. (Recall that an algorithm is *non-adaptive* if its queries do not depend on the answers to previous queries, or, equivalently, if all queries to the function can be made in parallel.) Furthermore, they showed that $\Omega(\log n)$ queries are necessary for non-adaptive algorithms. Subsequently, Fischer [66] showed that $\Omega(\log n)$ queries are necessary even for adaptive algorithms. Generalizing from monotonicity testing (when $k = 2$), Newman et al. gave in [109] the first sublinear-time algorithm for $(12\ldots k)$-freeness testing, whose query complexity is $(\log(n)/\varepsilon)^{O(k^2)}$. Their algorithm is non-adaptive and has one-sided error; as such, it outputs a length-k monotone subsequence with probability at least $9/10$ assuming the function f is ε-far from $(12\ldots k)$-free. However, other than the aforementioned lower bound of $\Omega(\log n)$ which follows from the case $k = 2$, no lower bounds were known for larger k.

The main result in this chapter settles the dependence on n in the query complexity of testing for $(12\ldots k)$-freeness with non-adaptive algorithms making one-sided error. Equivalently, we settle the complexity of non-adaptively finding a length-k monotone subsequence under the promise that the function $f\colon [n] \to \mathbb{R}$ is ε-far from $(12\ldots k)$-free.

Theorem 3.1. *Let $k \in \mathbb{N}$ be a fixed parameter. For any $\varepsilon > 0$, there exists an algorithm that, given query access to a function $f\colon [n] \to \mathbb{R}$ which is ε-far from $(12\ldots k)$-free, outputs a length-k monotone subsequence of f with probability at least $9/10$. The algorithm is non-adaptive and makes $(\log n)^{\lfloor \log_2 k \rfloor} \cdot \mathrm{poly}(1/\varepsilon)$ queries to f.*

Our algorithm thus significantly improves on the $(\log(n)/\varepsilon)^{O(k^2)}$-query non-adaptive algorithm of [109]. Furthermore, its dependence on n is optimal; in the full version of the results given here [22], we provide a matching lower bound of $\Omega((\log n)^{\lfloor \log_2 k \rfloor})$ on the non-adaptive query complexity. The lower bound holds even for the more restricted case where f is a permutation.

Related work Testing monotonicity of a function over a partially ordered set $\mathcal{X}$ is a well-studied and fruitful question, with works spanning the past two decades. Particular cases include when $\mathcal{X}$ is the line $[n]$ (see [18, 63, 66, 111] and Chapter 6), the Boolean hypercube $\{0,1\}^d$ [19, 35, 40, 45, 47, 48, 50, 52, 53, 95, 138], and the hypergrid $[n]^d$ [33, 36, 46]. We refer the reader to [81, Chapter 4] for further discussion on monotonicity testing.

This part of the thesis contributes to the line of work on finding order patterns in sequences and permutations. For the exact case, Guillemot and Marx [90] showed that an order pattern π of length k can be found in a sequence f of length n in time $2^{O(k^2 \log k)}n$; in particular, the problem of finding order patterns is fixed-parameter tractable (in the parameter k). Fox [74] later improved the running time to $2^{O(k^2)}n$. In the regime $k = \Omega(\log n)$, an algorithm of Berendsohn, Kozma, and Marx [28] running in time $n^{k/4+o(k)}$ provides the state-of-the-art. The analogous *counting* problem has also been actively studied, see [64] and the references within.

Two related questions are that of estimating the *distance to monotonicity* and the length of the *longest increasing subsequence* (LIS), which have also received significant attention from both the sublinear algorithms perspective [2, 113, 126], as well as the streaming perspective [62, 76, 87, 107, 125]. In particular, Saks and Seshadhri gave in [126] a randomized algorithm which, on input $f \colon [n] \to \mathbb{R}$, makes $\text{poly}(\log n, 1/\delta)$ queries and outputs $\widehat{m}$ approximating up to additive error δn the length of the longest increasing subsequence of f. This chapter and the following one also study monotone subsequences of the input function, albeit from a different (and incomparable) end of the problem. Loosely speaking, in [126] the main object of interest is a very *long* monotone subsequence (of length linear in n), and the task at hand is to get an estimate for its total length, whereas in our setting, there are $\Omega(n)$ disjoint copies of *short* monotone subsequences (of length k, which is a constant parameter), and these short subsequences may not necessarily combine to give one long monotone subsequence. Considering general permutations π of length k and *exact* computation, Guillemot and Marx [90] showed how to find a π-pattern in a sequence f in time $2^{O(k^2 \log k)}n$, later improved by Fox [74] to $2^{O(k^2)}n$.

3.2 Techniques

We now give a detailed overview of the techniques underlying Theorem 3.1, and provide some intuition behind the algorithms and notions we introduce. The starting point of our discussion will be the algorithm of Newman et al. [109], which we re-interpret in terms of the language used throughout this chapter; this will set up some of the main ideas behind our structural result (stated in Section 3.3), which will be crucial in the analysis of the algorithm. For simplicity, let $\varepsilon > 0$ be a small constant and let $k \in \mathbb{N}$ be fixed. Consider a function $f \colon [n] \to \mathbb{R}$ which is ε-far from $(12\dots k)$-free. This implies that there is a set $T \subseteq [n]^k$ of $\varepsilon n/k$ *disjoint* $(12\dots k)$-patterns. Specifically, the set T is comprised of k-tuples $(i_1, \dots, i_k) \in [n]$ where $i_1 < \cdots < i_k$ and $f(i_1) < \cdots < f(i_k)$ and each $i \in [n]$ appears in at most one k-tuple in T.[1] A key observation made in [109] is that if, for some $c \in [k-1]$, $(i_1, \dots, i_c, i_{c+1}, \dots, i_k)$ and $(j_1, \dots, j_c, j_{c+1}, \dots, j_k)$ are two k-tuples in T which

[1]To see why such T exists, take T to be a maximal set of disjoint $(12\dots k)$-patterns. Suppose $|T| < \varepsilon n/k$ and consider the function g given by greedily eliminating all $(12\dots k)$-patterns in f, and note that g is $(12\dots k)$-free and differs on f in less than εn indices.

satisfy $i_c < j_{c+1}$ and $f(i_c) < f(j_{c+1})$, then their combination

$$(i_1, \dots, i_c, j_{c+1}, \dots, j_k)$$

is itself a length-k monotone subsequence of f. Therefore, in order to design efficient sampling algorithms, one should analyze to what extent parts of different $(12\dots k)$-tuples from T may be combined to form length-k monotone subsequences of f.

Towards this goal, assign to each k-tuple $(i_1, \dots, i_k)$ in T a *distance profile* $\mathsf{dist\text{-}prof}(i_1, \dots, i_k) = (d_1, \dots, d_{k-1}) \in [\eta]^{k-1}$, where $\eta = O(\log n)$.[2] This distance profile is a $(k-1)$-tuple of non-negative integers satisfying

$$2^{d_j} \leq i_{j+1} - i_j < 2^{d_j+1} \qquad\qquad j \in [k-1]\,;$$

and let $\mathsf{gap}(i_1, \dots, i_k) = c \in [k-1]$ be the smallest integer where $d_c \geq d_j$ for all $j \in [k-1]$ (i.e., d_c denotes an (approximately) maximum length between two adjacent indices in the k-tuple). Suppose, furthermore, that for a particular $c \in [k-1]$, the subset $T_c \subseteq T$ of k-tuples whose gap is at c satisfies $|T_c| \geq \varepsilon n/k^2$ (such a $c \in [k-1]$ must exist since the T_c's partition T). If $(i_1, \dots, i_k) \in T_c$ and $\mathsf{dist\text{-}prof}(i_1, \dots, i_k) = (d_1, \dots, d_k)$, then the probability that a uniformly random element $\boldsymbol{\ell}$ of $[n]$ "falls" into that gap is

$$\mathop{\mathbf{Pr}}_{\boldsymbol{\ell} \sim [n]}\left[i_c \leq \boldsymbol{\ell} \leq i_{c+1}\right] \geq \frac{2^{d_c}}{n}. \tag{3.1}$$

Whenever this occurs for a particular k-tuple $(i_1, \dots, i_k)$ and $\boldsymbol{\ell} \in [n]$, we say that $\boldsymbol{\ell}$ *cuts* the tuple $(i_1, \dots, i_k)$. Note that the indices $i_{c+1}, \dots, i_k$ are contained within the interval $[\boldsymbol{\ell}, \boldsymbol{\ell} + k \cdot 2^{d_c+1}]$ and the indices $i_1, \dots, i_c$ are contained within the interval $[\boldsymbol{\ell} - k \cdot 2^{d_c+1}, \boldsymbol{\ell}]$. As a result, if we denote by $\delta_d(\boldsymbol{\ell}) \in [0,1]$, for each $d \in [\eta]$, the *density* of k-tuples from T_c lying inside $[\boldsymbol{\ell} - k \cdot 2^{d+1}, \boldsymbol{\ell} + k \cdot 2^{d+1}]$ (i.e., the fraction of this interval comprised of elements of T_c), we have

$$\mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim [n]}\left[\sum_{d \in [\eta]} \delta_d(\boldsymbol{\ell})\right] = \sum_{d \in [\eta]} \sum_{\substack{(i_1, \dots, i_k) \in T_c \\ \mathsf{dist\text{-}prof}(i_1, \dots, i_k)_c = d}} \mathop{\mathbf{Pr}}_{\boldsymbol{\ell} \sim [n]}\left[i_c \leq \boldsymbol{\ell} \leq i_{c+1}\right] \cdot \frac{1}{2 \cdot k \cdot 2^{d+1}} \gtrsim \frac{|T_c|}{n} \gtrsim \varepsilon. \tag{3.2}$$

For any ℓ achieving the above inequality, since $\eta = O(\log n)$, there exists some $d^* \in [\eta]$ such that $\delta_{d^*}(\ell) \gtrsim \varepsilon/\log n$. Consider now the set of k-tuples $T_{c,d^*}(\ell) \subseteq T_c$ contributing to $\delta_{d^*}(\ell)$, i.e., those k-tuples in T_c which are cut by ℓ and lie in $[\ell - k \cdot 2^{d^*+1}, \ell + k \cdot 2^{d^*+1}]$. Denote $r_{\text{med}} = \mathsf{median}\{f(i_c) : (i_1, \dots, i_k) \in T_{c,d^*}(\ell)\}$, and let

$$T_L = \left\{(i_1, \dots, i_c) : (i_1, \dots, i_k) \in T_{c,d^*}(\ell) \text{ and } f(i_c) \leq r_{\text{med}}\right\}, \qquad \text{and}$$
$$T_R = \left\{(i_{c+1}, \dots, i_k) : (i_1, \dots, i_k) \in T_{c,d^*}(\ell) \text{ and } f(i_c) \geq r_{\text{med}}\right\},$$

[2]We remark that the notion of a distance profile is solely used for the introduction and for explaining [109], and thus does not explicitly appear in subsequent sections.

where we note that T_L and T_R both have size at least $|T_{c,d^*}(\ell)|/2$. If the algorithm finds a c-tuple in T_L and a $(k-c)$-tuple in T_R, by the observation made in [109] that was mentioned above, the algorithm could combine the tuples to form a length-k monotone subsequence of f. At a high level, one may then recursively apply these considerations on $[\ell - k \cdot 2^{d^*+1}, \ell]$ with T_L and $[\ell, \ell + k \cdot 2^{d^*+1}]$ with T_R. A natural algorithm then mimics the above reasoning algorithmically, i.e., samples a parameter $\boldsymbol{\ell} \sim [n]$, and tries to find the unknown parameter $d^* \in [\eta]$ in order to recurse on both the left and right sides; once the tuples have length 1, the algorithm samples within the interval to find an element of T_L or T_R. This is, in essence, what the algorithm from [109] does, and this approach leads to a query complexity of $(\log n)^{O(k^2)}$. In particular, suppose that at each (recursive) iteration, the parameter c, corresponding to the gap of tuples in T, always equals 1. Note that this occurs when all $(12\dots k)$-patterns $(i_1,\dots,i_k)$ in T have $\textsf{dist-prof}(i_1,\dots,i_k) = (d_1,\dots,d_{k-1})$ with

$$d_1 \geq d_2 \geq \cdots \geq d_{k-1}. \tag{3.3}$$

Then, if k is at k_0, a recursive call leads to a set T_L containing 1-tuples, and T_R containing (k_0-1)-tuples. This only decreases the length of the subsequences needed to be found by 1 (so there will be $k-1$ recursive calls), while the algorithm pays for guessing the correct value of d^* out of $\Omega(\log n)$ choices, which may decrease the density of monotone k_0-subsequences within the interval of the recursive call by a factor as big as $\Omega(\log n)$.[3] As a result, the density of the length-k_0 monotone subsequence in the relevant interval could be as low as $\varepsilon/(\log n)^{k_0}$, which means that $(\log n)^{\Omega(k_0)}$ samples will be needed for the k_0-th round according to the above analysis, giving a total of $(\log n)^{\Omega(k^2)}$ samples (as opposed to $O((\log n)^{\lfloor \log_2 k \rfloor})$, which is the correct number, as we prove).

In order to overcome the above difficulty, we consider a particular choice of a family T of length-k monotone subsequences given by the "greedy" procedure (see Figure 3.1). Loosely speaking, this procedure begins with $T = \emptyset$ and iterates through each index $i_1 \in [n] \setminus T$. Each time, if (i_1) can be extended to a length-k monotone subsequence (otherwise it continues to the next available index), the procedure sets i_2 to be the first index, after i_1 and not already in T, such that (i_1, i_2) can be extended to a length-k monotone subsequence; then, it finds an index i_3 which is the next first index after i_2 and not in T such that (i_1, i_2, i_3) can be extended; and so on, until it has obtained a length-k monotone subsequence starting at i_1. It then adds the subsequence as a tuple to T, and repeats. This procedure eventually outputs a set T of disjoint, length-k monotone subsequences of f which has size at least $\varepsilon n/k^2$, and satisfies another crucial "interleaving" property (see Lemma 3.3):

($\star$) If $(i_1,\dots,i_k)$ and $(j_1,\dots,j_k)$ are k-patterns from T and $c \in [k-1]$ satisfy $j_1 < i_1$, $j_c < i_c$, and $i_{c+1} < j_{c+1}$, then $f(i_{c+1}) < f(j_{c+1})$.

[3]Initially, the density of T within $[n]$ is ε, and the density of T_L or T_R in $[\ell - k \cdot 2^{d^*+1}, \ell]$ and $[\ell, \ell + k \cdot 2^{d^*+1}]$ is $\varepsilon/\log n$.

Moreover, a slight variant of (3.1) guarantees that for any $(i_1, \ldots, i_k) \in T_c$ with $\textsf{dist-prof}(i_1, \ldots, i_k) = (d_1, \ldots, d_{k-1})$,

$$\mathop{\mathbf{Pr}}_{\boldsymbol{\ell} \sim [n]}\left[i_c + 2^{d_c}/3 \leq \boldsymbol{\ell} \leq i_{c+1} - 2^{d_c}/3\right] \gtrsim \frac{2^{d_c}}{n}.$$

Whenever the above event occurs, we say $\boldsymbol{\ell} \sim [n]$ *cuts* $(i_1, \ldots, i_k)$ at c *with slack*, and note that $i_1, \ldots, i_c$ lie in $[\boldsymbol{\ell} - k \cdot 2^{d_c+1}, \boldsymbol{\ell}]$ and $i_{c+1}, \ldots, i_k$ in $[\boldsymbol{\ell}, \boldsymbol{\ell} + k \cdot 2^{d_c+1}]$. We denote, similarly to the above, $\delta_d(\boldsymbol{\ell}) \in [0, 1]$ to be the density of k-tuples from T_c which are cut with slack by $\boldsymbol{\ell}$, and conclude (3.2). We then utilize $(\star)$ to make the following claim: suppose two k-tuples $(i_1, \ldots, i_k), (j_1, \ldots, j_k) \in T_c$ satisfy $\textsf{dist-prof}(i_1, \ldots, i_k) = (d_1, \ldots, d_{k-1})$, and $\textsf{dist-prof}(j_1, \ldots, j_k) = (d'_1, \ldots, d'_{k-1})$, where $d_c \leq d'_c - a \log k$, for some constant a which is not too small. If $(i_1, \ldots, i_k)$ and $(j_1, \ldots, j_k)$ are cut at c with slack, this means that ℓ lies roughly in the middle of i_c and i_{c+1} and of j_c and j_{c+1}, and since the distance between i_c and i_{c+1} is much smaller than that between j_c and j_{c+1}, the index j_1 will come before i_1, the index j_c will come before i_c, but the index i_{c+1} will come before j_{c+1}. By $(\star)$, $f(i_{c+1}) < f(j_{c+1})$ (cf. Lemma 3.14). In other words, the value, under the function f, of $(c + 1)$-th indices from tuples in $T_{c,d}(\ell)$ increases as d increases.

As a result, if $\ell \in [n]$ satisfies $\sum_{d \in [\eta]} \delta_d(\ell) \gtrsim \varepsilon$, and $\delta_d(\ell) \ll \varepsilon$ for all $d \in [\eta]$, that is, if the summands in (3.2) are *spread out*, an algorithm could find a length-k monotone subsequence by sampling, for many values of $d \in [\eta]$, indices which appear as the $(c+1)$-th index of tuples in $T_{c,d}(\ell)$. We call such values of ℓ the starts of *growing suffixes* (as illustrated in Figure 3.2). In Section 3.4.2, we describe an algorithm that makes $\tilde{O}(\log n/\varepsilon)$ queries and finds, with high probability, a length-k monotone subsequence if there are many such growing suffixes (see Lemma 3.20). The algorithm works by randomly sampling $\boldsymbol{\ell} \sim [n]$ and hoping that $\boldsymbol{\ell}$ is the start of a growing suffix; if it is, the algorithm samples enough indices from the segments $[\ell + 2^d, \ell + 2^{d+1}]$ to find a $(c + 1)$-th index of some tuple in $T_{c,d}(\ell)$, which gives a length-k monotone subsequence.

The other case corresponds to the scenario where $\ell \in [s]$ satisfies $\sum_{d \in [\eta]} \delta_d(\ell) \gtrsim \varepsilon$, but the summands are *concentrated* on few values of $d \in [\eta]$. In this case, we may consider a value of $d^* \in [\eta]$ which has $\delta_{d^*}(\ell) \gtrsim \varepsilon$, and then look at the intervals $[\ell - k \cdot 2^{d^*+1}, \ell]$ and $[\ell, \ell + k \cdot 2^{d^*+1}]$. We can still define T_L and T_R, both of which have size at least $|T_{c,d^*}|/2$ and have the property that any c-tuple from T_L can be combined with any $(k - c)$-tuple from T_R. Additionally, since $\delta_{d^*}(\ell) \gtrsim \varepsilon$, we crucially do *not* suffer a loss in the density of T_L and T_R in their corresponding intervals – a key improvement over the $\Omega(\log n)$ loss in density incurred by the original approach we first discussed. We refer to these intervals as *splittable intervals* (cf. Figure 3.3), and observe that they lead to a natural recursive application of these insights to the intervals $[\ell - k \cdot 2^{d^*+1}, \ell]$ and $[\ell, \ell + k \cdot 2^{d^*+1}]$. The main structural result, given in Theorem 3.11, does exactly this, and encodes the outcomes of the splittable intervals in an object we term a *k-tree descriptor* (see Section 3.3.3) whenever there are not too many growing suffixes. Intuitively, a k-tree descriptor consists of a rooted binary tree G on k leaves, as well as some additional information, which corresponds to a function $f \colon [n] \to \mathbb{R}$ without many growing suffixes. Each internal node v in G corresponds to a recursive application of the above insights, i.e., v has k_0 leaves in its subtree, a parameter $c_v \in [k_0 - 1]$ encoding the gap of sufficiently many k_0-tuples, and a collection of disjoint intervals of the form $[\ell - k \cdot 2^{d^*}, \ell + k \cdot 2^{d^*}]$ where ℓ cuts $(12 \ldots k_0)$-patterns with slack at c_v and satisfies (3.2); the left child of v has c leaves and contains the $(12 \ldots c)$-patterns in T_L and intervals $[\ell - k \cdot 2^{d^*}, \ell]$; the right child of v has $k_0 - c$ leaves and contains the $(12 \ldots (k_0 - c))$-patterns in T_R and intervals $[\ell, \ell + k \cdot 2^{d^*}]$ (see Figure 3.4).

The algorithm for this case is more involved than the previous, and leads to the $O((\log n)^{\lfloor \log_2 k \rfloor})$-query complexity stated in Theorem 3.1. The algorithm proceeds in $r_0 = 1 + \lfloor \log_2 k \rfloor$ rounds, maintaining a set $\mathbf{A} \subseteq [n]$, initially empty:

- *Round 1*: For each $i \in [n]$, include i in $\mathbf{A}$ independently with probability $\Theta(1/(\varepsilon n))$.
- *Round r, $2 \leq r \leq r_0$*: For each $i \in \mathbf{A}$ from the previous round, and each $j = 1, \ldots, O(\log n)$, consider the interval $B_{i,j} = [i - 2^j, i + 2^j]$. For each $i' \in B_{i,j}$, include i' in $\mathbf{A}$ independently with probability $\Theta(1/(\varepsilon 2^j))$.

At the end of all rounds, the algorithm queries f at all indices in $\mathbf{A}$, and outputs a $(12 \ldots k)$-pattern from $\mathbf{A}$, if one exists.

Recall the case considered in the sketch of the algorithm of [109], when the function f has all $(12 \ldots k)$-patterns $(i_1, \ldots, i_k)$ in T satisfying $\textsf{dist-prof}(i_1, \ldots, i_k) = (d_1, \ldots, d_{k-1})$ with $d_1 \geq d_2 \geq \ldots \geq d_{k-1}$. In this case, the k-tree descriptor G consists of a rooted binary tree of depth k. The root has a left child which is a leaf (corresponding to 1-tuples of first indices of some tuples in T, stored in T_L) and a right child (corresponding to suffixes of length $(k-1)$ of some tuples in T, stored in T_R) is an internal node. The root node corresponds to one application of the structural result, and the right child corresponds to a $(k-1)$-tree descriptor for the tuples in T_R. Loosely speaking, as $d_2 \geq \ldots \geq d_{k-1}$ the same reasoning repeats $k-1$ times, and leads to a path of length $k-1$ down the right children of the tree, the right child of the $(k-1)$-th internal node corresponding to a 1-tuple (i.e., a leaf).[4]

To gain some intuition, we analyze how the algorithm behaves on these instances. Suppose that in round 1, the algorithm samples an element $i \in [n]$ which is the k-th index of a 1-tuple stored in the right-most leaf of G. In particular, this index belongs to the set T_R of the $(k-1)$-th internal node, as a second index of a cut (12)-pattern in the $(k-1)$-th recursive call of the structural result. Similarly, i also belongs to that set T_R of the $(k-2)$-th internal node, as a part the third index of a cut (123)-pattern in the $(k-2)$-th recursive call. We may continue with all these inclusions to the root, i.e., i is the k-th element of some $(12 \ldots k)$-pattern in T, which is cut in the first call to the structural result. Round 2 of the algorithm will consider the $k-1$ intervals $B_{i,d'_{k-1}}, B_{i,d'_{k-2}}, \ldots, B_{i,d'_1}$, where $d'_j = d_j + \Theta(\log k)$, since it iterates through all $O(\log n)$ intervals of geometrically increasing

[4]This is somewhat inaccurate, as in each step, after forming T_L and T_R, we apply the greedy algorithm again and obtain new sets T'_L and T'_R, which may violate the assumption $d_1 \geq d_2 \geq \ldots \geq d_k$. We ignore this detail at the moment to simplify the explanation.

lengths.[5] One can check that for each $j \in [k-1]$, the interval B_{i,d'_j} contains $[\ell_j - k \cdot 2^{d_j}, \ell_j]$, where ℓ_j is some index which cut the $(k-j+1)$-tuple $(i_j, \dots, i_k)$ with slack in the j-th recursive call of the structural result. Recall that the set T_L of 1-tuples has density $\Omega(\varepsilon)$ inside $[\ell_j - k \cdot 2^{d_j}, \ell_j]$ and may be combined with any $(k-j)$-tuple from T_R. Following this argument, in the *second* round of the algorithm, $\mathbf{A}$ will include some index of T_L (for each $j \in [k-1]$), and these indices combine to form a $(12 \dots k)$-pattern – that is, with high probability, after two rounds, the algorithm succeeds in finding a monotone subsequence of length k.

Generalizing the above intuition for all possible distance profiles necessitates the use of $1 + \lfloor \log_2 k \rfloor$ rounds, and requires extra care. At a high level, consider an arbitrary k-tree descriptor G for $\Omega(\varepsilon n)$ many $(12 \dots k)$-patterns in f. Denote the root u, and consider the unique leaf w of G where the root-to-w path $(u_1, \dots, u_h)$ with $u_1 = u$ and $u_h = w$, satisfies that at each internal node u_l, the next node u_{l+1} is the child with larger number of leaves in its subtree.[6] We call such a leaf a *primary index* of G. The crucial property of the primary index is that the root-to-leaf path of w, $(u_1, u_2, \dots u_h)$, is such that the siblings of the nodes on this path[7] have strictly fewer than $k/2$ leaves in their subtrees.

The relevant event in the first round of the algorithm is that of sampling an index $i \in [n]$ which belongs to a 1-tuple of the primary index w of G. This occurs with probability at least $1 - 1/(100k)$, since we sample each element of $[n]$ with probability $\Theta(1/(\varepsilon n))$ while there are at least $\Omega(\varepsilon n)$ many $(12 \dots k)$-patterns. Now, roughly speaking, letting $(u_1, \dots, u_h)$ be the root-to-w path in G, and $(u'_2, \dots, u'_h)$ be the sibling nodes, the subtrees of G rooted at $u'_2, \dots, u'_h$ will be tree descriptors for the function f restricted to $B_{i,j}$'s and within these interval, the density of tuples is at least $\Omega(\varepsilon)$. As a result, the second round of the algorithm, recursively handles each subtree rooted at $u'_2, \dots, u'_h$ with one fewer round. Since the subtrees have strictly fewer than $k/2$ leaves, $\lfloor \log_2 k \rfloor - 1$ rounds are enough for an inductive argument. Moreover, since the total number of nodes is at most $2k$ and each recursive call succeeds with probability at least $1 - 1/(100k)$, by a union bound we may assume that all recursive calls succeed.

Unrolling the recursion, the query complexity $\Theta((\log n)^{\lfloor \log_2 k \rfloor})$ can be explained with a simple combinatorial game. We start with a rooted binary tree G on k leaves. In one round, whenever G is not simply a leaf, we pick the leaf w which is the primary index of G, and replace G with a collection of subtrees obtained by cutting out the root-to-w path in G. These rounds "pay" a factor of $\Theta(\log n)$, since the algorithm must find intervals on which the collection of subtrees form tree descriptors of f (restricted to these intervals). In the subsequent rounds, we recurse on each subtree simultaneously, picking the leaf of the primary index in each, and so on. After $\lfloor \log_2 k \rfloor$ many

[5]Note that the intervals B_{i,d'_j} and $B_{i,d'_{j+1}}$ may be the same, for instance when $d_j = d_{j+1}$.

[6]Ties are broken by picking the left child.

[7]For example, if $(u_1, \dots, u_h)$ is the root-to-w path where u_1 is the root and $u_h = w$, the sibling nodes along the path are given by $u'_2, u'_3, \dots, u'_h$, where u'_l is the sibling of u_l. Namely, if the l-th node on the root-to-w path is a left child of the $(l-1)$th node, then u'_l is the right child of the $(l-1)$-th node. Analogously, if the l-th node is a right child of the $(l-1)$-th node, then u'_l is the left child of the $(l-1)$-th node.

rounds, the trees are merely leaves, and the algorithm does not need to pay the factor $\Theta(\log n)$ to find good intervals, as it may simply sample from these intervals.

The execution of the above high-level plan is done in Section 3.4.3, where Lemma 3.21 is the main inductive lemma containing the analysis of the main algorithm (shown in Figure 3.8 and Figure 3.9).

Organization and notation We start by introducing the notation that we shall use throughout this chapter below. In Section 3.3 we prove our main structural result, and formally define the notions that underlie it: namely, Theorem 3.11, along with the definitions of growing suffixes and representation by tree descriptors (Definitions 3.6 and 3.10). Finally, Section 3.4 leverages this dichotomy to describe and analyze our testing algorithm, thus establishing the upper bound of Theorem 3.1 (see Theorem 3.19 for a formal statement).

We write $a \lesssim b$ if there exists a universal positive constant $C > 0$ such that $a \leq Cb$, and $a \asymp b$ if we have both $a \lesssim b$ and $b \lesssim a$. We frequently denote $\mathcal{I}$ as a collection of disjoint intervals, $I_1, \ldots, I_s$, and then write $\mathcal{S}(\mathcal{I})$ for the set of all sub-intervals which lie within some interval in $\mathcal{I}$. For two collections of disjoint intervals $\mathcal{I}_0$ and $\mathcal{I}_1$, we say that $\mathcal{I}_1$ is a *refinement* of $\mathcal{I}_0$ if every interval in $\mathcal{I}_1$ is contained within an interval in $\mathcal{I}_0$. (We remark that it is not the case that intervals in $\mathcal{I}_1$ must form a partition of intervals in $\mathcal{I}_0$.) For a particular set $A \subseteq [n]$ and an interval $I \subseteq [n]$, we define the *density* of A in I as the ratio $|A \cap I|/|I|$. Given a set S, we write $\boldsymbol{x} \sim S$ to indicate that $\boldsymbol{x}$ is a random variable given by a sample drawn uniformly at random from S, and $\mathcal{P}(S)$ for the power set of S. Given a sequence f of length n, we shall interchangeably use the notions $(12\ldots k)$-*copy* (or $(1, 2, \ldots, k)$-copy), $(12\ldots k)$-*pattern*, and *length-k increasing subsequence*, to refer to a tuple of elements $x_1 < \ldots < x_k \in [n]$ such that $f(x_1) < \ldots < f(x_k)$.

3.3 Structural Result

3.3.1 Rematching Procedure

Let $f\colon [n] \to \mathbb{R}$ be a function which is ε-far from $(12\ldots k)$-free. Let T be a set of k-tuples representing monotone subsequences of length k within f, i.e.,

$$T \subseteq \left\{ (i_1, \ldots, i_k) \in [n]^k : i_1 < \cdots < i_k \text{ and } f(i_1) < \cdots < f(i_k) \right\},$$

and for such T let $E(T)$ be the set of indices of subsequences in T, so

$$E(T) = \bigcup_{(i_1,\ldots,i_k)\in T} \{i_1, \ldots, i_k\}.$$

Observation 3.2. *If $f\colon [n] \to \mathbb{R}$ is ε-far from $(12\ldots k)$-free, then there exists a set $T \subseteq [n]^k$ of disjoint length-k increasing subsequences of f such that $|T| \geq \varepsilon n/k$.*

To see why the observation holds, take T to be a maximal disjoint set of such k-tuples. Then we can obtain a $(12\dots k)$-free sequence from f by changing only the entries of $E(T)$ (e.g. for every $i \in E(T)$ define $f(i) = f(j)$ where j is the largest $[n] \setminus E(T)$ which is smaller than i. If there is no $j \in [n] \setminus E(T)$ where $j < i$, let $f(i) = \max_{\ell \in [n]} f(\ell)$). Since f is ε-far from being $(12\dots k)$-free, we have $|E(T)| \geq \varepsilon n$, thus $|T| \geq \varepsilon n/k$.

In this section, we show that from a function $f\colon [n] \to \mathbb{R}$ which is ε-far from $(12\dots k)$-free and a set T_0 of disjoint, length-k monotone subsequences of f, a greedy rematching algorithm finds a set T of disjoint, length-k monotone subsequences of f where $E(T) \subseteq E(T_0)$ with some additional structure, which will later be exploited in the structural lemma and the algorithm. The greedy rematching algorithm, `GreedyDisjointTuples`, is specified in Figure 3.1; for convenience, in view of its later use in the algorithm, we phrase it in terms of an arbitrary parameter k_0, not necessarily the (fixed) parameter k itself.

Lemma 3.3. *Let $k_0 \in \mathbb{N}$, $f\colon [n] \to \mathbb{R}$, and let $T_0 \subseteq [n]^{k_0}$ be a set of disjoint monotone subsequences of f of length k_0. Then there exists a set $T \subseteq [n]^{k_0}$ of disjoint k_0-tuples with $E(T) \subseteq E(T_0)$ such that the following holds.*

1. *The set T holds disjoint monotone subsequences of length k_0.*
2. *The size of T satisfies $|T| \geq |T_0|/k_0$.*
3. *For any two $(i_1, \dots, i_{k_0}), (j_1, \dots, j_{k_0}) \in T$ and any $\ell \in [k_0 - 1]$, if $i_1 < j_1$, $i_\ell < j_\ell$ and $i_{\ell+1} > j_{\ell+1}$ then $f(i_{\ell+1}) > f(j_{\ell+1})$.*

Proof of Lemma 3.3. We show that the subroutine `GreedyDisjointTuples`(f, k_0, T_0), described in Figure 3.1, finds a set T with $E(T) \subseteq E(T_0)$ satisfying properties 1, 2, and 3. Property 1 is clear from the description of `GreedyDisjointTuples`(f, k_0, T_0). For 2, suppose $|T| < |T_0|/k_0$, then, there exists a tuple $(i_1, \dots, i_{k_0}) \in T_0$ with $\{i_1, \dots, i_{k_0}\} \cap E(T) = \emptyset$. Since `GreedyDisjointTuples`$(f, k_0, T_0)$ increases the size of T throughout the execution, $\{i_1, \dots, i_{k_0}\} \cap T = \emptyset$ at every point in the execution of the algorithm. This is a contradiction; when $i = i_1$, a monotone subsequence disjoint from T would have been found, and i_1 included in T. Finally, for 3, consider the iteration when $i = i_1$, and note that at this moment, $T \cap \{i_1, \dots, i_{k_0}, j_1, \dots, j_{k_0}\} = \emptyset$. Suppose that $i_\ell < j_\ell$, $j_{\ell+1} < i_{\ell+1}$; if $f(j_{\ell+1}) \geq f(i_{\ell+1})$, then $(i_1, \dots, i_\ell, j_{\ell+1}, \dots, j_{k_0})$ is an increasing subsequence in $E(T_0) \setminus E(T)$, which means that $j_{\ell+1}$ would have been preferred over $i_{\ell+1}$, a contradiction. □

Definition 3.4 (c-gap)**.** *Let $(i_1, \dots, i_{k_0})$ be a monotone subsequence of f and let $c \in [k_0 - 1]$. We say that $(i_1, \dots, i_{k_0})$ is a* c-gap subsequence *if c is the smallest integer such that $i_{c+1} - i_c \geq i_{b+1} - i_b$ for all $b \in [k_0 - 1]$.*

Note that for a set T of disjoint length-k_0 monotone subsequences of f, we may partition the k_0-tuples of T into $(T_1, \dots, T_{k_0 - 1})$ where for each $c \in [k_0 - 1]$, T_c holds the c-gap monotone

```
Subroutine GreedyDisjointTuples $(f, k_0, T_0)$

Input: A function $f\colon [n] \to \mathbb{R}$, integer $k_0 \in \mathbb{N}$, and a set $T_0$ of disjoint monotone subsequences of $f$ of length $k_0$.
Output: a set $T \subseteq [n]^{k_0}$ of disjoint monotone subsequences of $f$ of length $k_0$.

  1. Let $T = \emptyset$ and $i$ be the minimum element in $E(T_0)$. Repeat the following.
       i. Let $i_1 \leftarrow i$. If there exists $j_2, \dots, j_{k_0} \in E(T_0) \setminus E(T)$ such that $(i_1, j_2, \dots, j_{k_0})$
          is an increasing subsequence of $f$, pick $i_2, \dots, i_{k_0} \in E(T_0) \setminus E(T)$ recursively as
          follows: for $\ell = 2, \dots, k_0$, let $i_\ell$ be the smallest element in $E(T_0) \setminus E(T)$ for which
          there exist $j_{\ell+1}, \dots, j_{k_0} \in E(T_0) \setminus E(T)$ such that $(i_1, \dots, i_\ell, j_{\ell+1}, \dots, j_{k_0})$ is an
          increasing subsequence of $f$.
      ii. If $(i_1, \dots, i_{k_0})$ is a monotone subsequence found by (i), set $T \leftarrow T \cup \{(i_1, \dots, i_{k_0})\}$.
     iii. Let $i$ be the next element of $E(T_0) \setminus E(T)$, if such an element exists; otherwise,
          proceed to 2.
  2. Output $T$.
```

Figure 3.1: Description of the `GreedyDisjointTuples` subroutine.

subsequences of T. As these sets form a partition of T, the following lemma is immediate from Lemma 3.3.

Lemma 3.5. *Let $f\colon [n] \to \mathbb{R}$, and let T_0 be a set of disjoint length-k_0 monotone subsequences of f. Then there exist $c \in [k_0 - 1]$ and a family $T \subseteq [n]^{k_0}$ of disjoint monotone subsequences of f, with $E(T) \subseteq E(T_0)$ such that the following holds.*

1. *The subsequences in T are all c-gap subsequences.*
2. *$|T| \geq |T_0|/k_0^2$.*
3. *For any two $(i_1, \dots, i_{k_0}), (j_1, \dots, j_{k_0}) \in T$ and any $\ell \in [k_0 - 1]$, if $i_1 < j_1$, $i_\ell < j_\ell$ and $i_{\ell+1} > j_{\ell+1}$ then $f(i_{\ell+1}) > f(j_{\ell+1})$.*

3.3.2 Growing Suffixes and Splittable Intervals

We now proceed to set up notation and prepare for the main structural theorem for sequences $f\colon [n] \to \mathbb{R}$ which are ε-far from $(12\dots k)$-free. In order to simplify the presentation of the subsequent discussion, consider fixed $k \in \mathbb{N}$ and $\varepsilon \in (0,1)$, as well as a fixed sequence $f\colon [n] \to \mathbb{R}$ which is ε-far from $(12\dots k)$-free. By Observation 3.2 and Lemma 3.5, there exists an integer $c \in [k-1]$

and a set T of disjoint monotone subsequences of f which have a c-gap, satisfying $|T| \geq \varepsilon n/\operatorname{poly}(k)$ and property 3 from Lemma 3.5. For the rest of the subsection, we consider a fixed setting of such $c \in [k-1]$ and set T.

We show (in Theorem 3.8) that one of the following two possibilities holds. Either there is a large set of what we call *growing suffixes* (see Definition 3.6 for a formal definition), or there are disjoint intervals which we call *splittable* (see Definition 3.7 for a formal definition). Intuitively, a growing suffix will be given by the suffix $(a, n]$ and will have the property that by dividing $(a, n]$ into $\Theta(\log_2(n-a))$ segments of geometrically increasing lengths, there are many monotone subsequences $(i_1, \dots, i_k)$ of f lying inside $(a, n]$ where each i_t belongs to a different segment. In the other case, an interval $[a, b]$ is called splittable if it can be divided into three sub-intervals of roughly equal size, which we refer to as the left, middle, and right intervals, with the following property: the left interval contains a large set T_L of $(12\dots c)$-patterns, the right interval contains a large set T_R of $(12\dots(k-c))$-patterns, and combining any $(12\dots c)$-pattern in T_L with any $(12\dots(k-c))$-pattern in T_R yields a $(12\dots k)$-pattern.

For each index $a \in [n]$, let $\eta_a = \lceil \log_2(n-a) \rceil$. Let $S_1(a), \dots, S_{\eta_a}(a) \subseteq [n]$ be disjoint intervals given by $S_t(a) = [a+2^{t-1}, a+2^t) \cap [n]$. The collection of intervals $S(a) = (S_t(a) : t \in [\eta_a])$ partitions the suffix $(a, n]$ into intervals of geometrically increasing lengths (except possibly the last interval, which may be shorter), and we refer to the collection $S(a)$ as the *growing suffix* at a.

Definition 3.6. *Let $\alpha, \beta \in [0, 1]$. We say that an index $a \in [n]$ starts an* (α, β)-growing suffix *if, when considering the collection of intervals $S(a) = \{S_t(a) : t \in [\eta_a]\}$, for each $t \in [\eta_a]$ there is a subset $D_t(a) \subseteq S_t(a)$ of indices such that the following properties hold.*

1. *We have $|D_t(a)|/|S_t(a)| \leq \alpha$ for all $t \in [\eta_a]$, and $\sum_{t=1}^{\eta_a} |D_t(a)|/|S_t(a)| \geq \beta$.*
2. *For every $t, t' \in [\eta_a]$ where $t < t'$, if $b \in D_t(a)$ and $b' \in D_{t'}(a)$, then $f(b) < f(b')$.*

Intuitively, our parameter regime will correspond to the case when α is much smaller than β, specifically, $\alpha \leq \beta/\operatorname{poly}(k)$, for a sufficiently large-degree polynomial in k. If $a \in [n]$ starts an (α, β)-growing suffix with these parameters, then the η_a segments, $S_1(a), \dots, S_{\eta_a}(a)$, contain many monotone subsequences of length k which are algorithmically easy to find (given access to the start a). Indeed, by (2), it suffices to find a k-tuple $(i_1, \dots, i_k)$ such that $i_1 \in D_{t_1}, \dots, i_k \in D_{t_k}$, for some $t_1, \dots, t_k \in [\eta_a]$ with $t_1 < \dots < t_k$ (see Figure 3.2). By (1), the sum of densities is at least β, yet each density is less than $\alpha \leq \beta/\operatorname{poly}(k)$. In other words, the densities of $D_1(a), \dots, D_{\eta_a}(a)$ within $S_1(a), \dots, S_{\eta_a}(a)$, respectively, must be spread out, which implies, intuitively, that there are many ways to pick suitable $i_1, \dots, i_k$.

Definition 3.7. *Let $\alpha, \beta \in (0, 1]$ and $c \in [k_0 - 1]$. Let $I \subseteq [n]$ be an interval, let $T \subseteq I^{k_0}$ be a set of disjoint, length-k_0 monotone subsequences of f lying in I, and define*

$$T^{(L)} = \{(i_1, \dots, i_c) \in I^c : (i_1, \dots, i_c) \text{ is a prefix of a } k_0\text{-tuple in } T\}, \text{ and}$$
$$T^{(R)} = \{(j_1, \dots, j_{k_0-c}) \in I^{k_0-c} : (j_1, \dots, j_{k_0-c}) \text{ is a suffix of a } k_0\text{-tuple in } T\}.$$

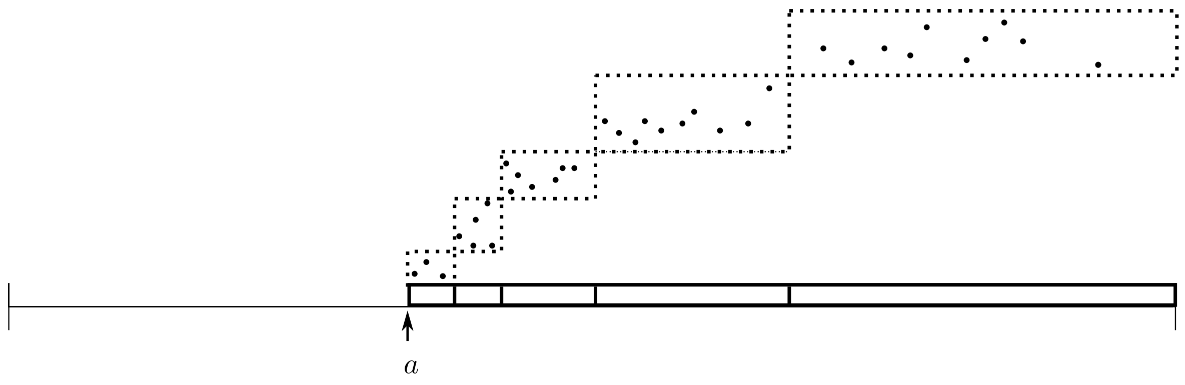


Figure 3.2: Growing Suffixes. Depiction of an (α, β)-growing suffix at index $a \in [n]$ (see Definition 3.6). The labeled segments $S_t(a)$ are shown, as well as the subsets $D_t(a)$. Notice that for all j, all the elements in $D_t(a)$ lie below those in $D_{t+1}(a)$. In Section 3.4.2, we show that if an algorithm knows that a starts an (α, β)-growing suffix, for $\alpha \le \beta/\operatorname{poly}(k)$, then sampling $\operatorname{poly}(k)/\beta$ many random indices from each $S_t(a)$ finds a monotone pattern with probability at least 0.9.

*We say that the pair (I, T) is (c, α, β)-*splittable *if $|T|/|I| \ge \beta$; $f(i_c) < f(j_1)$ for every $(i_1, \dots, i_c) \in T^{(L)}$ and $(j_1, \dots, j_{k_0-c}) \in T^{(R)}$; and there is a partition of I into three adjacent intervals $L, M, R \subseteq I$ (that appear in this order, from left to right) of size at least $\alpha|I|$, satisfying $T^{(L)} \subseteq L^c$ and $T^{(R)} \subseteq R^{k_0-c}$.*

*A collection of disjoint interval-tuple pairs $(I_1, T_1), \dots, (I_s, T_s)$ is called a (c, α, β)-*splittable collection of T *if each (I_j, T_j) is (c, α, β)-splittable and the sets $(T_j : j \in [s])$ partition T.*

We now state the main theorem of this section, whose proof will be given in Section 3.3.5.

Theorem 3.8. *Let $k, k_0 \in \mathbb{N}$ be positive integers satisfying $1 \le k_0 \le k$, and let $\delta \in (0, 1)$ and let $C > 0$. Let $f : [n] \to \mathbb{R}$ be a function and let $T_0 \subseteq [n]^{k_0}$ be a set of δn disjoint monotone subsequences of f of length k_0. Then there exists an $\alpha \ge \Omega(\delta/k^5)$ such that at least one of the following conditions holds.*

1. *Either there exists a set $H \subseteq [n]$, of indices that start an $(\alpha, Ck\alpha)$-growing suffix, satisfying $\alpha|H| \ge \delta n/\operatorname{poly}(k, \log(1/\delta))$; or*

2. *There exists an integer c with $1 \le c < k_0$, a set T, with $E(T) \subseteq E(T_0)$, of disjoint length-k_0 monotone subsequences, and a $(c, 1/(6k), \alpha)$-splittable collection of T, of disjoint interval-tuple pairs $(I_1, T_1), \dots, (I_s, T_s)$, such that*

$$\alpha \sum_{h=1}^{s} |I_h| \ge \frac{|T_0|}{\operatorname{poly}(k, \log(1/\delta))}.$$

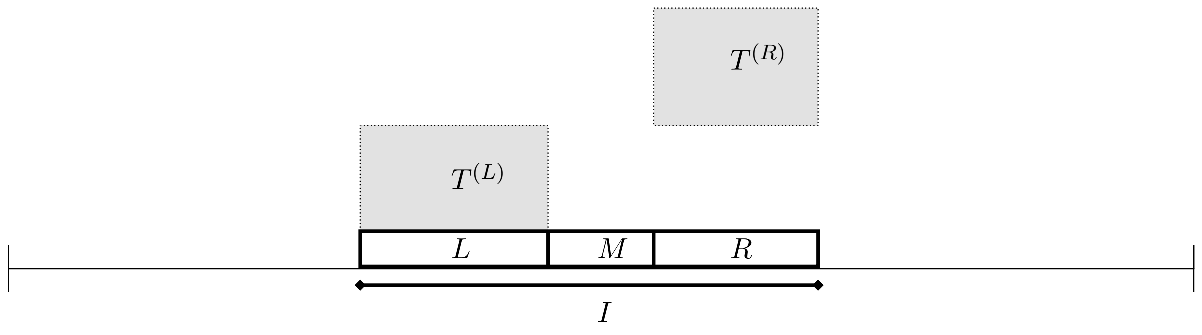


Figure 3.3: Splittable Intervals. Depiction of a (c, α, β)-splittable interval, as defined in Definition 3.7. The interval I is divided into three adjacent intervals, L, M, and R, and the disjoint monotone sequences are divided so that $T^{(L)}$ contains the indices $(i_1, \dots, i_c)$ and $T^{(R)}$ contains the indices $(i_{c+1}, \dots, i_k)$. Furthermore, we have that every $(i_1, \dots, i_c) \in T^{(L)}$ and $(j_{c+1}, \dots, j_k) \in T^{(R)}$ have $f(i_c) < f(j_{c+1})$, so that any monotone pattern of length c in $E(T^{(L)})$ may be combined with any monotone pattern of length $k - c$ in $E(T^{(R)})$ to obtain a monotone pattern of length k within I.

We remark that the above theorem is stated with respect to the two parameters, k_0 and k, for ease of applicability. In particular, in the next section, we will apply Theorem 3.8 multiple times, and it will be convenient to have k be fixed and k_0 be a varying parameter. In that sense, even though the monotone subsequences in question have length k_0, the relevant parameters that Theorem 3.8 lower bounds only depend on k.

Consider the following scenario: $f : [n] \to \mathbb{R}$ is a sequence which is ε-far from $(12 \dots k)$-free, so by Observation 3.2, there exists a set T_0 of disjoint, length-k monotone subsequences of f of size at least $\varepsilon n/k$. Suppose that upon applying Theorem 3.8 with $k_0 = k$ and $\delta = \varepsilon/k$, (2) holds. Then, there exists a $(c, 1/(6k), \alpha)$-splittable collection of a large subset of disjoint, length-k monotone subsequences T into disjoint interval-tuple pairs $(I_1, T_1), \dots, (I_s, T_s)$. For each $h \in [s]$, the pair (I_h, T_h) is $(c, 1/(6k), \alpha)$-splittable, so let $I_h = L_h \cup M_h \cup R_h$ be the left, middle, and right intervals of I_h; furthermore, let $T_h^{(L)}$ be the $(12 \dots c)$-patterns in L_h which appear as prefixes of T_h, and $T_h^{(R)}$ be the $(12 \dots (k-c))$-patterns in R_h which appear as suffixes of T_h in R_h. Thus, the restricted function $f_{|L_h} : L_h \to \mathbb{R}$ contains $|T_h|$ disjoint $(12 \dots c)$-patterns, and $f_{|R_h} : R_h \to \mathbb{R}$ contains $|T_h|$ disjoint $(12 \dots (k-c))$-patterns. This naturally leads to a recursive application of Theorem 3.8 to the function $f_{|L_h}$ with $k_0 = c$, and to the function $f_{|R_h}$ with $k_0 = k - c$, for all $h \in [s]$.

3.3.3 Tree Descriptors

We now introduce the notion of *tree descriptors*, which will summarize information about a function f after applying Theorem 3.8 recursively. Then, we state the main structural result for functions that are ε-far from $(12 \dots k)$-free. The goal is to say that every function which is ε-far from

$(12\dots k)$-free either has many growing suffixes, or there exists a tree descriptor which describes the behavior of many disjoint, length-k monotone subsequences in the function. The following two definitions make up the notion of a tree descriptor representing a function. Figure 3.4 shows an example of Definitions 3.9 and 3.10.

Definition 3.9. *Let $k_0 \in \mathbb{N}$ and $\delta \in (0,1)$. A (k_0,δ)-*weighted-tree *is a pair (G,ϱ), where*

- *$G = (V, E, w)$ is a rooted binary tree with edges labeled by a function $w\colon E \to \{0,1\}$. Every non-leaf node has two outgoing edges, e_0, e_1 with $w(e_0) = 0$ and $w(e_1) = 1$. The set of leaves $V_\ell \subseteq V$ satisfies $|V_\ell| = k_0$, and $\leq_G$ is the total order defined on the leaves by the values of w on a root-to-leaf path.*[8]

- *$\varrho\colon V \to [\lceil \log(1/\delta) \rceil]$ is a function that assigns a positive integer to each node of G.*

In the next definition, we show how we use weighted trees to represent a function f and a set of disjoint, length-k_0 monotone subsequences.

Definition 3.10. *Let $k, k_0 \in \mathbb{N}$ be such that $1 \leq k_0 \leq k$, let $\alpha \in (0,1)$, let $I \subseteq \mathbb{N}$ be an interval, and let $f\colon I \to \mathbb{R}$ be a function. Let $T \subseteq I^{k_0}$ be a set of disjoint monotone subsequences of f. A triple (G, ϱ, I) is called a (k, k_0, δ)-*tree descriptor[9] *of (f, T, I), if (G,ϱ) is a (k_0,δ)-weighted tree, I is a function $\mathsf{I}\colon V \to \mathcal{P}(\mathcal{I})$ (where $V = V(G)$), and the following recursive definition holds.*

1. *If $k_0 = 1$ (so $T \subseteq I$),*

   - *The graph $G = (V, E, w)$ is the rooted tree with one node, r, and no edges.*
   - *The function $\varrho\colon V \to [\lceil \log(1/\delta) \rceil]$ (simply mapping one node) satisfies $2^{-\varrho(r)} \leq |T|/|I| \leq 2^{-\varrho(r)+1}$.*
   - *The map $\mathsf{I}\colon V \to \mathcal{S}(I)$ is given by $\mathsf{I}(r) = \{\{t\} : t \in T\}$.*

2. *If $k_0 > 1$,*

   - *The graph $G = (V, E, w)$ is a rooted binary tree with k_0 leaves. We refer to the root by r, the left child of the root (namely, the child incident with the edge given 0 by w) by v_{left}, and the right child of the root (the child incident with the edge given 1) by v_{right}. Let c be the number of leaves in the subtree of v_{left}, so v_{right} has $k_0 - c$ leaves in its subtree.*
   - *Write $\mathsf{I}(r) = \{I_1, \dots, I_s\}$. Then $I_1, \dots, I_s$ are disjoint sub-intervals of I, and, setting $T_i = (I_i)^{k_0} \cap T$, the pairs $(I_1, T_1), \dots, (I_s, T_s)$ form a $(c, 1/(6k), 2^{-\varrho(r)})$-splittable collection of T, and*
   $$2^{-\varrho(r)} \sum_{h=1}^{s} |I_h| \geq \frac{|T|}{\mathrm{poly}(k, \log(1/\delta))^k}.$$

[8]Specifically, for $l_1, l_2 \in V_\ell$ at depths d_1 and d_2, with root to leaf paths $(r, u^{(1)}, \dots, u^{(d_1-1)}, l_1)$ and $(r, v^{(1)}, \dots, v^{(d_2-1)}, l_2)$, then $l_1 \leq_G l_2$ if and only if $(w(r,u^{(1)}), w(u^{(1)},u^{(2)}), \dots, w(u^{(d_1-1)}, l_1)) \leq (w(r,v^{(1)}), w(v^{(1)},v^{(2)}), \dots, w(v^{(d_2-1)}, l_2))$ in the natural partial order on $\{0,1\}^*$.

[9]We shall sometimes refer to this as a k_0-tree descriptor, in particular when k, δ are not crucial to the discussion.

- *For each $h \in [s]$ there exists a partition (L_h, M_h, R_h) of I_h that satisfies Definition 3.7, such that the sets $T_h^{(L)}$, of prefixes of length c of subsequences in T_h, and $T_h^{(R)}$, of suffixes of length $k_0 - c$ of subsequences in T_h, satisfy $T_h^{(L)} \subseteq (L_h)^c$ and $T_h^{(R)} \subseteq (R_h)^{k_0-c}$. Moreover, the following holds.*
  The tuple $(G_{\mathsf{left}}, \varrho_{\mathsf{left}}, \mathsf{I}_{h,\mathsf{left}})$ is a (k, c, δ)-tree descriptor of f, $T_h^{(L)}$, and L_h, where G_{left} is the subtree rooted at v_{left}, ϱ_{left} is the restriction of ϱ to the subtree G_{left}, and $\mathsf{I}_{h,\mathsf{left}}$ is defined by $\mathsf{I}_{h,\mathsf{left}}(v) := \{J \in \mathsf{I}(v) \colon J \subseteq L_h\}$ for all $v \in G_{\mathsf{left}}$.
  Analogously, the tuple $(G_{\mathsf{right}}, \varrho_{\mathsf{right}}, \mathsf{I}_{h,\mathsf{right}})$ is a $(k, k_0 - c, \delta)$-tree descriptor of f, $T_h^{(R)}$, and R_h, where $G_{\mathsf{right}}, \varrho_{\mathsf{right}}, \mathsf{I}_{h,\mathsf{right}}$ are defined analogously.

We remark that it is *not* the case that for every function $f \colon I \to \mathbb{R}$ defined on an interval I, and for every $T \subseteq I^{k_0}$ which is a set of disjoint, length-k_0 monotone subsequences of f, there must exist a k_0-tree descriptor which represents (f, T, I). The goal will be to apply Theorem 3.8 recursively whenever we are in (2), and to find a sufficiently large set T of disjoint length-k monotone subsequences, as well as a k-tree descriptor which represents (f, T, I).

3.3.4 The Structural Dichotomy Theorem

We are now in a position to state the main structural theorem of far-from-$(12 \dots k)$-free sequences, which guarantees that every far-from-$(12 \dots k)$-free sequence either has many growing suffixes, or can be represented by a tree descriptor. The algorithm for finding a $(12 \dots k)$-pattern will proceed by considering the two cases independently. The first case, when a sequence has many growing suffixes, is easy for algorithms; we will give a straight-forward sampling algorithm making roughly $O_k(\log n/\varepsilon)$ queries. The second case, when a sequence is represented by a tree descriptor is the "hard" case for the algorithm.

Theorem 3.11 (Main structural result)**.** *Let $k \in \mathbb{N}$, $\varepsilon > 0$, and let $f \colon [n] \to \mathbb{R}$ be a function which is ε-far from $(12 \dots k)$-free. Then one of the following holds, where $C > 0$ is a large constant.*

- *There exists a parameter $\alpha \geq \varepsilon / \operatorname{poly}(k, \log(1/\varepsilon))^k$, and a set $H \subseteq [n]$ of indices which start an $(\alpha, Ck\alpha)$-growing suffix, with*
$$\alpha |H| \geq \frac{\varepsilon n}{\operatorname{poly}(k, \log(1/\varepsilon))^k},$$
- *or there exists a set $T \subseteq [n]^k$ of disjoint monotone subsequences of f satisfying*
$$|T| \geq \frac{\varepsilon n}{\operatorname{poly}(k, \log(1/\varepsilon))^{k^2}}$$
  and a (k, k, β)-tree descriptor (G, ϱ, I) representing $(f, T, [n])$ where $\beta \geq \varepsilon / \operatorname{poly}(k, \log(1/\varepsilon))^{k^2}$.

Proof. We shall prove the following claim, by induction, for all $k_0 \in [k]$. Here $C > 0$ is a large constant, and $C' > 0$ is a large enough constant such that $\alpha \geq \delta/(C'k^5)$ in the statement of Theorem 3.8, applied with the constant C.

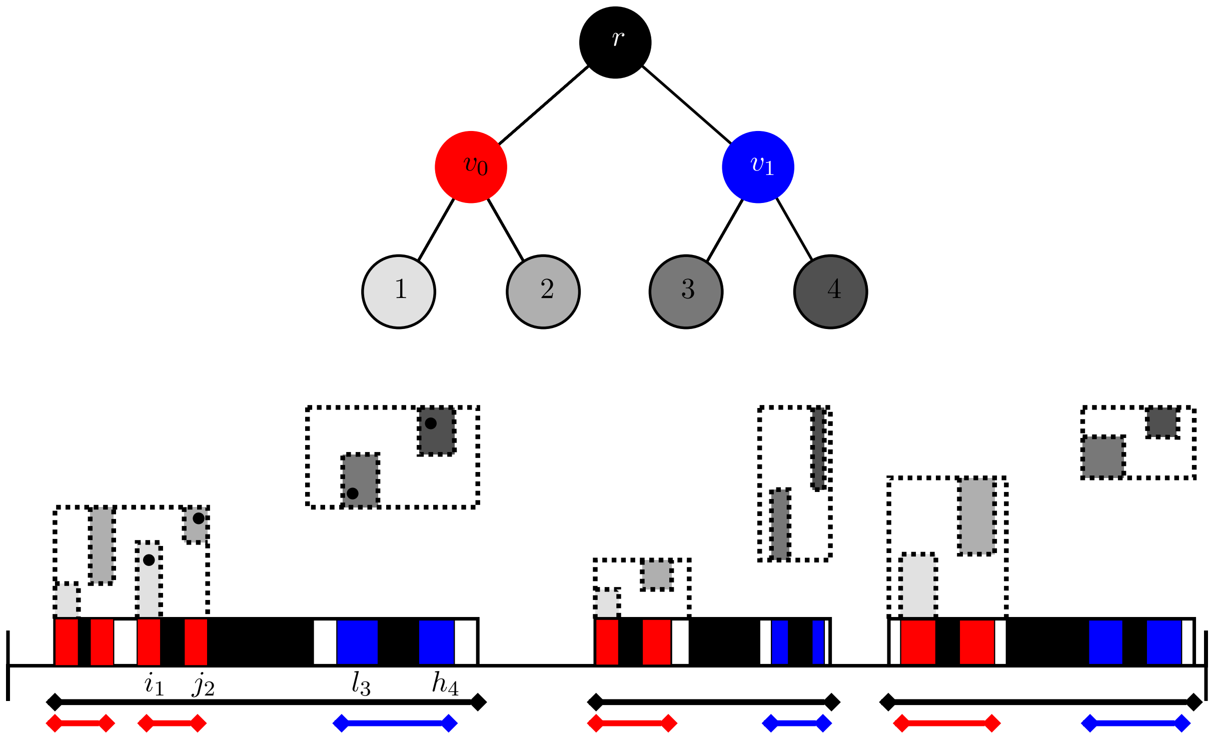


Figure 3.4: Tree Descriptors. Depiction of a tree descriptor (G, ϱ, I) representing (f, T, I), as described in Definitions 3.9 and 3.10. The graph G displayed above is a rooted tree with four leaves, which are ordered and labeled left-to-right. The root node r, filled in black, has its corresponding intervals from $\mathsf{I}(r)$ shown below the sequence as three black intervals. Each of the black intervals in $\mathsf{I}(r)$ is a $(2, \alpha, \beta)$-splittable interval, for $\alpha \approx 1/3$ and $\beta \geq 1/6$. Then, the root has the left child v_0, filled in red, and the right child v_1, filled in blue. The red intervals are those belonging to $\mathsf{I}(v_0)$, and the blue intervals are those belonging to $\mathsf{I}(v_1)$. Each black interval in $\mathsf{I}(r)$ has a left part, which contains intervals in $\mathsf{I}(v_0)$, and a right part, which contains intervals in $\mathsf{I}(v_1)$. The red and blue intervals in $\mathsf{I}(v_0)$ and $\mathsf{I}(v_1)$ are also $(1, \alpha, \beta)$-splittable, and the left part of the red intervals contains indices which will form the 1 in the monotone pattern of length 4, and the right part of the red intervals contains indices which will form the 2. Likewise, the left part of blue intervals will contain the indices corresponding to 3, and the right part of the blue intervals will contain indices corresponding to 4. The regions where the indices from T lie are shown above the sequence, where the indices 1–4 of some monotone pattern in T lie in regions which are progressively darker. In order to see how a monotone subsequence may be sampled given that (G, ℓ, I) is a tree descriptor for (f, T, I) with sufficiently large T, consider indices i_1 and j_2 that belong to some subsequences from T, and lie in different shaded regions of the same red interval, within a black interval; and furthermore, l_3 and h_4 belong to some subsequence from T, and lie in different shaded regions of the same blue interval, within the same black interval as i_1 and j_2; then, the subsequence (i_1, j_2, l_3, h_4) is a monotone subsequence even though $(i_1, j_2, l_3, h_4) \notin T$.

Claim. Let $K = C'k^5$ and let $P(\cdot,\cdot)$ be the function from the statement of Theorem 3.8; so $P(x,y) = \text{poly}(x, \log y)$, and we may assume that P is increasing in both variables. Let $A(\cdot,\cdot)$ and $B(\cdot,\cdot)$ be increasing functions, such that

$$\begin{aligned}
&A(k_0, 1/\delta) \geq 12k\lceil \log(K^{k_0}/\delta)\rceil \cdot P(k, 1/\delta) \cdot A(k_0 - 1, K/\delta) \\
&A(1, 1/\delta) = 1/\delta \\
&B(k_0, 1/\delta) \geq 2 \cdot P(k, K/\delta) \cdot (2k\lceil \log(KB(k_0-1, K/\delta)/\delta)\rceil)^{2k_0} \cdot B(k_0 - 1, K/\delta) \\
&B(1, 1/\delta) = 1/\delta
\end{aligned} \tag{3.4}$$

Note that there exists such $A(\cdot,\cdot)$ and $B(\cdot,\cdot)$ with $A(k, 1/\delta) = (\text{poly}(k, \log(1/\delta)))^k$ and $B(k, 1/\delta) = (\text{poly}(k, \log(1/\delta)))^{k^2}$.

Let $I \subseteq \mathbb{N}$ be an interval, let g be a sequence $g\colon I \to \mathbb{R}$, let $T_0 \subseteq I^{k_0}$ be a set of disjoint length-k_0 monotone subsequences, and define $\delta := |T_0|/|I|$. Then

1. Either there exists $\alpha \geq \delta/K^{k_0}$, which is an integer power of $1/2$, along with a set $H \subseteq I$ of $(\alpha, Ck\alpha)$-growing suffix start points such that

$$\alpha|H| \geq \frac{\delta|I|}{A(k_0, 1/\delta)},$$

2. Or there exists a set $T \subseteq I^{k_0}$ of disjoint k_0-tuples satisfying $E(T) \subseteq E(T_0)$ and

$$|T| \geq \frac{|T_0|}{B(k_0, 1/\delta)}$$

and a (k, k_0, α)-tree descriptor (G, ϱ, I) for (g, T, I), where $\alpha \geq \delta/B(k_0, 1/\delta)$.

Note that since f is ε-far from $(12\dots k)$-free, there is a set $T_0 \subseteq [n]^k$ of at least $\varepsilon n/k$ disjoint length-k monotone subsequences. By applying the above claim for $k_0 = k$, T_0, $[n]$ and f, the theorem follows. Thus, it remains to prove the claim; we proceed by induction.

if $k_0 = 1$: Note that here T_0 is a subset of I. We define the $(k, 1, \delta)$-tree descriptor (G, ϱ, I) which represents $f, T = T_0, I$ in the natural way:

- $G = (V, E)$ is a rooted tree with one node: $V = \{r\}$ and $E = \emptyset$.
- $\varrho\colon V \to \mathbb{N}$ is given by $\varrho(r) = \lceil \log(1/\delta)\rceil$, so $2^{-\varrho(r)} \leq |I \cap T|/|I| \leq 2^{-\varrho(r)+1}$.
- $\mathsf{I}\colon V \to \mathcal{S}(I)$ is given by $\mathsf{I}(r) = \{\{t\} : t \in T\}$.

if $2 \leq k_0 \leq k$: By Theorem 3.8, there exists $\alpha \geq \delta/K$ such that one of (1) and (2), from the statement of the theorem, holds.

- If (1) holds, there is a set $H \subseteq I$ of $(\alpha, Ck\alpha)$-growing suffix start points with

$$\alpha|H| \geq \frac{\delta|I|}{P(k, 1/\delta)};$$

note that we may assume that α is an integer power of $1/2$.[10]

- Otherwise, (2) holds, and we are given an integer $c \in [k_0 - 1]$, a set T of disjoint length-k_0 monotone subsequences, with $E(T) \subseteq E(T_0)$, and a $(c, 1/(6k), \alpha)$-splittable collection of T into disjoint interval-tuple pairs $(I_1, T_1), \ldots, (I_s, T_s)$, such that

$$\alpha \sum_{h=1}^{s} |I_h| \geq \frac{|T_0|}{P(k, 1/\delta)} = \frac{\delta |I|}{P(k, 1/\delta)}.$$

  Recall that by definition of splittability, $|T_h|/|I_h| \geq \alpha$ for every $h \in [s]$.

If (1) holds, we are done; so we assume that (2) holds.

For each $h \in [s]$, since (I_h, T_h) is a $(c, 1/(6k), \alpha)$-splittable pair, there exists a partition (L_h, M_h, R_h) that satisfies the conditions stated in Definition 3.7. Let $T_h^{(L)}$ be the collection of prefixes of length c of subsequences in T_h, and let $T_h^{(R)}$ be the collection of suffixes of length $k_0 - c$ of subsequences in T_h.

We apply the induction hypothesis to each of the pairs $(L_h, T_h^{(L)})$ and $(R_h, T_h^{(R)})$. We consider two cases for each $h \in [s]$.

1. (1) holds for either $(L_h, T_h^{(L)})$ or $(R_h, T_h^{(R)})$. This means that there exists β_h, which is an integer power of $1/2$, and which satisfies $\beta_h \geq \alpha / K^{\max\{c, k_0 - c\}} \geq \alpha / K^{k_0 - 1} \geq \delta / K^{k_0}$, and a set $H_h \subseteq I_h$ of start points of $(\beta_h, Ck\beta_h)$-growing subsequences, such that (using $|R_h|, |L_h| \geq |I_h|/(6k)$)
$$\beta_h |H_h| \geq \frac{\alpha |I_h|}{6k \cdot A(k_0 - 1, 1/\alpha)}$$

2. Otherwise, (2) holds for both $(L_h, T_h^{(L)})$ and $(R_h, T_h^{(R)})$. Setting $\beta = \alpha / B(k_0 - 1, 1/\alpha)$, this means that there exists a (k, c, β)-tree descriptor $(G_h^{(L)}, \varrho_h^{(L)}, \mathsf{l}_h^{(L)})$, for $(g, \mathcal{L}_h, L_h)$ where $\mathcal{L}_h \subseteq (L_h)^c$ is a set of length-c monotone subsequences, such that $E(\mathcal{L}_h) \subseteq E(T_h^{(L)})$ and
$$|\mathcal{L}_h| \geq \frac{|T_h^{(L)}|}{B(k_0 - 1, 1/\alpha)}, \tag{3.5}$$
   and, similarly, there exists a $(k, k_0 - c, \beta)$-tree descriptor $(G_h^{(R)}, \varrho_h^{(R)}, \mathsf{l}_h^{(R)})$ for $(g, \mathcal{R}_h, L_h)$, where $\mathcal{R}_h \subseteq (R_h)^{k_0 - c}$ is a set of length-$(k_0 - c)$ monotone subsequences, such that $E(\mathcal{R}_h) \subseteq E(T_h^{(R)})$ and
$$|\mathcal{R}_h| \geq \frac{|T_h^{(R)}|}{B(k_0 - 1, 1/\alpha)}. \tag{3.6}$$
   For convenience, we shall assume that $|\mathcal{L}_h| = |\mathcal{R}_h|$, by possibly removing some elements of the largest of the two (and reflecting this in the corresponding tree descriptor).

[10]to be precise and to ensure that we can take α to be an integer power of 2, it might be better to apply Theorem 3.8 with constant $2C$, to allow for some slack; this does not change the argument.

Suppose first that

$$\sum_{h\colon \text{ first case holds for } h} |I_h| \geq \frac{1}{2} \cdot \sum_{h=1}^{s} |I_h|.$$

Since each β_h is an integer power of $1/2$, there are at most $\lceil \log(K^{k_0}/\delta) \rceil$ possible values for β_h. Hence, there exists some β (with $\beta \geq \delta/K^{k_0}$) such that the collection S, of indices $h \in [s]$ for which the first case holds for h and $\beta_h = \beta$, satisfies

$$\sum_{h \in S} |I_h| \geq \frac{1}{2\lceil \log(K^{k_0}/\delta) \rceil} \cdot \sum_{h=1}^{s} |I_h|.$$

Let $H = \bigcup_{h \in S} H_h$. Then H is a set of start points of $(\beta, Ck\beta)$-growing suffixes, with

$$\begin{aligned}\beta|H| &\geq \frac{\alpha}{6k \cdot A(k_0-1, 1/\alpha)} \cdot \sum_{h \in S} |I_h| \geq \frac{\alpha}{12k\lceil \log(K^{k_0}/\delta) \rceil \cdot A(k_0-1, 1/\alpha)} \cdot \sum_{h=1}^{s} |I_h| \\ &\geq \frac{\delta|I|}{12k\lceil \log(K^{k_0}/\delta) \rceil \cdot P(k, 1/\delta) \cdot A(k_0-1, 1/\alpha)} \geq \frac{\delta|I|}{A(k_0, 1/\delta)},\end{aligned}$$

where the last inequality follows from (3.4). This proves the claim in this case.

Next, we may assume that

$$\sum_{h\colon \text{ second case holds for } h} |I_h| \geq \frac{1}{2} \cdot \sum_{h=1}^{s} |I_h|.$$

Note that the number of quadruples $(G_h^{(L)}, \varrho_h^{(L)}, G_h^{(R)}, \varrho_h^{(R)})$ (whose elements are as above) is at most $(2c)^{2c}(2(k_0-c))^{2(k_0-c)}(\lceil \log(1/\beta) \rceil)^{2k_0} \leq (2k\lceil \log(1/\beta) \rceil)^{2k_0}$, since the number of trees on l vertices is at most l^l, and we have at most $\lceil \log(1/\beta) \rceil$ possible weights to assign to each of the vertices. It follows that there exists such a quadruple $(G_L^*, \varrho_L^*, G_R^*, \varrho_R^*)$ such that if S is the set of indices h that were assigned this quadruple, then

$$\begin{aligned}\alpha \cdot \sum_{h \in S} |I_h| &\geq \frac{\alpha}{(2k\lceil \log(1/\beta) \rceil)^{2k_0}} \cdot \sum_{\text{second case holds for } h} |I_h| \\ &\geq \frac{\alpha}{2 \cdot (2k\lceil \log(1/\beta) \rceil)^{2k_0}} \cdot \sum_{h=1}^{s} |I_h| \geq \frac{|T_0|}{2 \cdot P(k, 1/\delta) \cdot (2k\lceil \log(1/\beta) \rceil)^{2k_0}}.\end{aligned} \tag{3.7}$$

We form a set $\mathcal{T}_h$ of monotone length-k_0 subsequences by matching elements from $\mathcal{L}_h$ with elements from $\mathcal{R}_h$ for each $h \in S$; that they can be matched follows from the assumption that $|\mathcal{L}_h| = |\mathcal{R}_h|$, and that these form monotone subsequences follows from the assumptions on $\mathcal{L}_h, \mathcal{R}_h$. Set $\mathcal{T} := \cup_{h \in S} \mathcal{T}_h$. Note that $(I_h, \mathcal{T}_h)$ is (k_0, c, β)-splittable by (3.5) and (3.6) (using $\beta = \alpha/B(k_0-1, 1/\alpha)$). Let (G, ϱ) be the (k, k_0, β)-weighted-tree obtained by taking a root r, with weight $\varrho(r) = \lceil \log(1/\beta) \rceil$, adding the tree (G_L^*, ϱ^*) as a subtree to its left (i.e., the root of this tree is joined to r by an edge with value 0) and adding the tree (G_R^*, ϱ^*) as a subtree to its right. Now, we form a (G, ϱ, I)-tree descriptor by setting

$$\mathsf{I}(v) = \begin{cases} \{I_h : h \in S\} & v = r \\ \bigcup_{h \in S} \mathsf{I}_h^{(L)}(v) & v \in G_L^* \\ \bigcup_{h \in S} \mathsf{I}_h^{(R)}(v) & v \in G_R^*. \end{cases}$$

We claim that (G, ϱ, I) is a (k, k_0, β)-tree descriptor for $(g, \mathcal{T}, I)$. Indeed, $((I_h, \mathcal{T}_h))_{h \in S}$ is a $(c, 1/(6k), 2^{-\varrho(r)})$-splittable collection of $\mathcal{T}$, and, by (3.7) and because $|T_0| \geq |\mathcal{T}|$

$$2^{-\varrho(r)} \sum_{h \in S} |I_h| \geq \frac{\alpha}{2} \cdot \sum_{h \in S} |I_h| \geq \frac{|\mathcal{T}|}{4 \cdot P(k, 1/\delta) \cdot (2k\lceil \log(1/\beta) \rceil)^{2k_0}} = \frac{|\mathcal{T}|}{\mathrm{poly}(k, \log(1/\delta))^k}.$$

The remaining requirements in the recursive definition of a tree descriptor (see Definition 3.10) follow as $(G_L^*, \varrho^*, \mathsf{I}_h^{(L)})$ is a (k, c, β)-tree descriptor for $(g, \mathcal{L}_h, L_h)$ and $(G_L^*, \varrho^*, \mathsf{I}_h^{(R)})$ is a $(k, k_0 - c, \beta)$-tree descriptor for $(g, \mathcal{R}_h, R_h)$ for every $h \in S$. Since $\beta = \alpha/B(k_0 - 1, 1/\alpha) \geq \delta/B(k_0, 1/\delta)$, it follows that (G, ϱ, I) is a $(k, k_0, \delta/B(k_0, 1/\delta))$-tree descriptor for $(g, \mathcal{T}, I)$.

It remains to lower-bound the size of $\mathcal{T}$. Using (3.6) and (3.7), we have

$$\begin{aligned} |\mathcal{T}| = \sum_{h \in S} |\mathcal{R}_h| &\geq \frac{1}{B(k_0 - 1, 1/\alpha)} \cdot \sum_{h \in S} |\mathcal{T}_h| \geq \frac{\alpha}{B(k_0 - 1, 1/\alpha)} \cdot \sum_{h \in S} |I_h| \\ &\geq \frac{|T_0|}{2 \cdot P(k, 1/\delta) \cdot (2k\lceil \log(1/\beta) \rceil)^{2k_0} \cdot B(k_0 - 1, 1/\alpha)} \geq \frac{|T_0|}{B(k_0, 1/\delta)}. \end{aligned}$$

This completes the proof of the inductive claim in this case. □

3.3.5 Proof of Structural Dichotomy Theorem

We now prove Theorem 3.8. For the rest of this section, let $k, k_0 \in \mathbb{N}$, with $1 \leq k_0 \leq k$, be fixed, and let $f \colon [n] \to \mathbb{R}$ be a fixed function. Let T_0 be a set of δn disjoint monotone subsequences of f of length k_0. We apply Lemma 3.5 to the set T_0; this specifies an integer $c \in [k_0 - 1]$ and a subset T of at least $\delta n/k^2$ disjoint monotone subsequences of length k_0 satisfying the conclusion of Lemma 3.5.

Definition 3.12. *Let $(i_1, \ldots, i_{k_0}) \in [n]^{k_0}$ be a monotone subsequence with a c-gap. We say that $(i_1, \ldots, i_{k_0})$ is* at scale t *if $2^t \leq i_{c+1} - i_c \leq 2^{t+1}$, where $t \in \{0, \ldots, \lfloor \log n \rfloor\}$.*

Definition 3.13. *Let $(i_1, \ldots, i_{k_0}) \in [n]^{k_0}$ be a monotone subsequence with a c-gap. For $\gamma \in (0, 1)$, we say that $\ell \in [n]$* γ-cuts $(i_1, \ldots, i_{k_0})$ at c with slack *if*

$$i_c + \gamma(i_{c+1} - i_c) \leq \ell \leq i_{c+1} - \gamma(i_{c+1} - i_c). \tag{3.8}$$

We hereafter consider the parameter setting of $\gamma := 1/3$. For $\ell \in [n]$, $t \in \{0, \dots, \lfloor \log n \rfloor\}$, and any subset $U \subset T$ of disjoint $(12 \dots k_0)$-patterns in f let

$$A_t(\ell, U) = \{(i_1, \dots, i_{k_0}) \in U : (i_1, \dots, i_{k_0}) \text{ is at scale } t \text{ and is } \gamma\text{-cut at } c \text{ with slack by } \ell\}. \quad (3.9)$$

We note that for each $(i_1, \dots, i_{k_0}) \in A_t(\ell, U)$, the index i_{c+1} is in $[\ell, \ell + 2^{t+1}]$, and since $A_t(\ell, U)$ is made of disjoint monotone sequences, $|A_t(\ell, U)| \le 2^{t+1}$.

Lemma 3.14. *For every $\ell \in [n]$, $t \in \{0, \dots, \lfloor \log n \rfloor\}$, and $U \subset T$,*

- *Every $(i_1, \dots, i_{k_0}) \in A_t(\ell, U)$ satisfies*
$$\ell - (k-1)2^{t+1} \le i_1, \dots, i_c \le \ell - \gamma 2^t \qquad \ell + \gamma 2^t \le i_{c+1}, \dots, i_{k_0} \le \ell + (k-1)2^{t+1}.$$
- *Let $t_1 \ge t_2 + 1 + \log(1/\gamma) + \log(c+1)$, $(i_1, \dots, i_{k_0}) \in A_{t_1}(\ell, U)$ and $(j_1, \dots, j_{k_0}) \in A_{t_2}(\ell, U)$. Then $f(j_{c+1}) < f(i_{c+1})$.*

Proof. Fix any $\ell \in [n]$, $t \in \{0, \dots, \lfloor \log n \rfloor\}$ and $U \subset T$. To establish the first bullet, consider any $(i_1, \dots, i_{k_0}) \in A_t(\ell, U)$. By definition of a c-gap sequence, we have

$$i_1 \ge i_{c+1} - c(i_{c+1} - i_c) \ge \ell - (k-1)2^{t+1},$$

using $i_{c+1} - i_c \le 2^{t+1}$ and $i_{c+1} \ge \ell$. By (3.8), we have $i_c \le \ell - \gamma 2^t$ (using $i_{c+1} - i_c \ge 2^t$). The first inequality follows as $i_1 < \dots < i_c$. The inequality for $i_{c+1}, \dots, i_{k_0}$ follows similarly.

For the second bullet, let $(i_1, \dots, i_{k_0}) \in A_{t_1}(\ell, U)$ and $(j_1, \dots, j_{k_0}) \in A_{t_2}(\ell, U)$ and suppose that $2^{t_1} \ge 2^{t_2+1} \cdot (c+1)/\gamma$. We have $i_c \le \ell - \gamma 2^{t_1}$ and $j_c \ge \ell - 2^{t_2+1}$ (using (3.8) and (3.9)), from which it follows that $j_c > i_c$. Similarly, $i_1 < i_c \le \ell - \gamma 2^{t_1}$ and $j_1 \ge \ell - (c-1)2^{t_2+1}$, implying that $j_1 > i_1$, and $i_{c+1} \ge \ell + \gamma 2^{t_1}$ and $j_{c+1} \le \ell + 2^{t_2+1}$, which implies that $i_{c+1} > j_{c+1}$. The inequality $f(j_{c+1}) < f(i_{c+1})$ follows from the assumption that T satisfies (3) from Lemma 3.5. ☐

The proof of Theorem 3.8 follows by considering a random $\boldsymbol{\ell} \sim [n]$ and the collection of sets $A_1(\boldsymbol{\ell}, T), \dots, A_{\lfloor \log n \rfloor}(\boldsymbol{\ell}, T)$. By looking at how the sizes of the sets $A_1(\boldsymbol{\ell}, T), \dots, A_{\log n - 1}(\boldsymbol{\ell}, T)$ vary, we will be able to say that $\boldsymbol{\ell}$ is the start of a growing suffix, or identify a splittable interval. Towards this goal, we first establish a simple lemma; here $v(\ell, U)$ is defined to be $\sum_{t=0}^{\lfloor \log n \rfloor} |A_t(\ell, U)|/2^t$.

Lemma 3.15. *Let $U \subset T$ be any subset and $\boldsymbol{\ell} \sim [n]$ be sampled uniformly at random. Then*

$$\mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim [n]} v(\ell, U) \ge \frac{|U|}{3n}.$$

Proof. Fix a sequence $i = (i_1, \dots, i_{k_0}) \in U$, and let $t(i) \in \{0, \dots, \lfloor \log n \rfloor\}$ be its scale. Then, the probability (over a uniformly random $\boldsymbol{\ell}$ in $[n]$) that i belongs to $A_{t(i)}(\boldsymbol{\ell}, U)$ is lower bounded as

$$\mathop{\mathbf{Pr}}_{\boldsymbol{\ell} \sim [n]}[i \in A_{t(i)}(\boldsymbol{\ell}, U)] \ge \frac{(1-2\gamma)2^{t(i)}}{n} = \frac{2^{t(i)}}{3n}.$$

Therefore, $\sum_{t=0}^{\log n-1} \sum_{i \in U \,:\, t(i)=t} \mathbf{Pr}_{\boldsymbol{\ell} \sim [n]}[i \in A_t(\boldsymbol{\ell}, U)]/2^t \geq |U|/(3n)$, or, equivalently, since $\mathbf{Pr}_{\boldsymbol{\ell} \sim [n]}[i \in A_t(\boldsymbol{\ell}, U)] = 0$ for $t \neq t(i)$,

$$\mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim [n]}\left[\sum_{t=0}^{\log n-1} \frac{|A_t(\boldsymbol{\ell}, U)|}{2^t}\right] = \mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim [n]}\left[\sum_{t=0}^{\log n-1} \sum_{i \in U} \frac{\mathbb{1}\{i \in A_t(\boldsymbol{\ell}, U)\}}{2^t}\right] \geq \frac{|U|}{3n},$$

establishing the lemma. □

We next establish an auxiliary lemma that we will use in order to find growing suffixes.

Lemma 3.16. *Let $\ell \in [n]$ and $U \subset T$ be such that every $t \in \{0, \dots, \lfloor \log n \rfloor\}$ satisfies $|A_t(\ell, U)|/2^t \leq \beta$. Then, if $\ell' \in [n]$ is any index satisfying*

$$\max\{i_c : (i_1, \dots, i_{k_0}) \in A_t(\ell, U), t \in \{0, \dots, \lfloor \log n \rfloor\}\} \leq \ell' \leq \ell, \tag{3.10}$$

then ℓ' is the start of an $(4\beta, v(\ell, U)/(12 \log k))$-growing suffix.

Proof. Let $\Delta = 1 + \log(1/\gamma) + \log(c+1)$, and notice that $3 \leq \Delta \leq 3 \log k$. Then, there exists a set $\mathcal{T} \subseteq \{0, \dots, \lfloor \log n \rfloor\}$ such that

1. All distinct $t, t' \in \mathcal{T}$ satisfy $|t - t'| \geq \Delta$; and,
2. $\sum_{t \in \mathcal{T}} \frac{|A_t(\ell,U)|}{2^t} \geq \frac{1}{\Delta+1} \sum_{t=0}^{\log n-1} \frac{|A_t(\ell,U)|}{2^t} = \frac{v(\ell,U)}{\Delta+1}$.

(Such a set exists by an averaging argument.) Now, consider the sets

$$D_t(\ell) = \begin{cases} \{i_{c+1} : (i_1, \dots, i_{k_0}) \in A_t(\ell, U)\} & \text{if } t \in \mathcal{T} \\ \emptyset & \text{if } t \in \{0, \dots, \lfloor \log n \rfloor\} \setminus \mathcal{T}. \end{cases}$$

Considering any $\ell' \in [n]$ satisfying (3.10), we have the following for all $t \in \{0, \dots, \lfloor \log n \rfloor\}$ with $D_t(\ell) \neq \emptyset$: $\ell - 2^{t+1} \leq \ell' \leq \ell$; $\min D_t(\ell) \geq \ell + 2^t/3$; and $\max D_t(\ell) \leq \ell' + 2^{t+1}$. Therefore, $D_t(\ell) \subset S_{t-1}(\ell') \cup S_t(\ell') \cup S_{t+1}(\ell')$. (Recall that $S_t(a) = [a + 2^{t-1}, a + 2^t)$.) For each $t \in \mathcal{T}$, let $n(t) \in \{t-1, t, t+1\}$ satisfying $|D_t(\ell) \cap S_{n(t)}(\ell')| \geq |D_t(\ell)|/3$, and notice that all $n(t) \in \{0, \dots, \lfloor \log n \rfloor\}$ are distinct since $\Delta \geq 3$.

The first condition in Definition 3.6 holds as the densities of $D_t(\ell) \cap S_{n(t)}(\ell')$ in the corresponding intervals $S_{n(t)}(\ell')$ are upper bounded by $|D_t(\ell)|/|S_{n(t)}(\ell')| \leq |A_t(\ell, U)|/2^{t-2} \leq 4\beta$, and the sum of these densities satisfies

$$\sum_{t \in \mathcal{T}} \frac{|D_t(\ell) \cap S_{n(t)}(\ell')|}{|S_{n(t)}(\ell')|} \geq \sum_{t \in \mathcal{T}} \frac{|D_t(\ell)|}{3 \cdot 2^t} = \sum_{t \in \mathcal{T}} \frac{|A_t(\ell, U)|}{3 \cdot 2^t} \geq \frac{v(\ell, U)}{3(\Delta+1)},$$

which is at least $v(\ell, U)/(12 \log k)$. The second condition in Definition 3.6 holds, because for any choice of $b \in D_t(\ell), b' \in D_{t'}(\ell)$ with $t < t'$, we have $t' \geq t + \Delta$ (by the choice of $\mathcal{T}$), and hence $f(b) < f(b')$ by the second item of Lemma 3.14. □

Lemma 3.17. *For every $\eta > 0$, there exists a subset $U \subset T$ such that every $(i_1, \ldots, i_{k_0}) \in U$ has i_c as the start of an $(1, \eta)$-growing suffix, and every $\ell \in [n]$ satisfies $v(\ell, T \setminus U) \leq 12\eta \log(k)$.*

Proof. Define sets U_j, elements ℓ_j, and k_0-tuples $(i_{j,1}, \ldots, i_{j,k_0})$ recursively as follows. Set $U_0 := \emptyset$, and given a set U_{j-1}, if $v(\ell, T \setminus U_{j-1}) \leq 12\eta \log k$ for every $\ell \in [n]$, stop; otherwise, let $\ell_j \in [n]$ be such that $v(\ell_j, T \setminus U_j) > 12\eta \log k$ and define $U_j = U_{j-1} \cup \{(i_{j,1}, \ldots, i_{j,k_0})\}$, where

$$i_{j,c} = \max\{i_c : (i_1, \ldots, i_{k_0}) \in T \setminus U_j \text{ and } (i_1, \ldots, i_{k_0}) \text{ is } \gamma\text{-cut by } \ell_j\}.$$

Let j^* be the maximum j for which U_j was defined, and set $U := U_{j^*}$. Every k_0-tuple in U is of the form $(i_{j,1}, \ldots, i_{j,k_0})$ for some $j \leq j^*$. By Lemma 3.16, applied with $\ell = \ell_j$, $U = T \setminus U_{j-1}$, $i_{j,c}$, it follows that $i_{j,c}$ is the start of an $(1, \eta)$-growing suffix, for every j for which U_j was defined. Lemma 3.17 follows. □

We let $C > 0$ be a large enough constant. Let $U \subset T$ be the set obtained from Lemma 3.17 with $\eta = Ck$, and suppose that $|U| \geq |T|/2$. Then, we may let $\alpha = 1$ and $H = \{i_c : (i_1, \ldots, i_{k_0}) \in U\}$. Notice that every index in H is the start of an $(\alpha, Ck\alpha)$-growing suffix, and since $|H| \geq |T|/2$, we obtain the first item in Theorem 3.8. Suppose then, that $|U| < |T|/2$, and consider the set $V = T \setminus U$. By definition of V, we now have $v(\ell, V) \leq 12Ck \log k$ for every $\ell \in [n]$. Let b_0 be the largest integer which satisfies $2^{b_0} \leq 12Ck \log k$ and b_1 be the smallest integer which satisfies $2^{-b_1} \leq \delta/(12k^2)$, so $2^{b_0} \lesssim 2^{b_1} \asymp k^2/\delta$. For $-b_0 \leq j \leq b_1$, consider the pairwise-disjoint sets

$$B_j = \left\{\ell \in [n] : 2^{-j} \leq v(\ell, V) \leq 2^{-j+1}\right\}, \tag{3.11}$$

and note that by Lemma 3.15, since $|V| \geq |T|/2 \geq \delta n/2k^2$,

$$\frac{1}{n} \sum_{j=-b_0}^{b_1} |B_j| \cdot 2^{-j+1} \geq \frac{1}{n} \sum_{\ell \in [n]} v(\ell, V) \geq \frac{\delta}{6k^2}.$$

Thus, denoting

$$\mu := \frac{\delta}{6k^2(b_1 + b_0 + 1)} \asymp \frac{\delta}{k^2 \log(k/\delta)},$$

there is an integer $-b_0 \leq j^* \leq b_1$ that satisfies

$$|B_{j^*}| \cdot 2^{-j^*} \geq \mu n. \tag{3.12}$$

Lemma 3.18. *There exists a deterministic algorithm,* `GreedyDisjointIntervals`(f, B, j)*, which takes three inputs: a function $f \colon [n] \to \mathbb{R}$, a set $B \subseteq [n]$ of integers, and an integer $j \in [-b_0, b_1]$, and outputs a collection $\mathcal{I}$ of interval-tuple pairs or a subset $H \subseteq B$. An execution of the algorithm* `GreedyDisjointIntervals`(f, B_{j^*}, j^*) *where μ, B_{j^*} and j^* are defined in* (3.12)*, satisfies one of the following two conditions, where $C > 0$ is a large constant.*

- *The algorithm returns a set $H \subseteq B$ of indices that start a $(4 \cdot 2^{-j^*}/(Ck \log k), 2^{-j^*}/(12 \log k))$-growing suffix, and $|H| \geq 2^{j^*-1}\mu n$; or*

- *The algorithm returns a* $(c, 1/(6k), 2^{-j^*}/(8Ck^2\log k))$*-splittable collection* $(I_1, T_1), \ldots, (I_s, T_s)$*, where* $\sum_{h=1}^{s} |I_h| \geq 2^{j^*-2}\mu n$*.*

```
Subroutine GreedyDisjointIntervals($(f, B, j)$)

Input: A function $f\colon [n] \to \mathbb{R}$, a set $B \subseteq [n]$ and an integer $j$, such that every $\ell \in B$ satisfies
$2^{-j} \leq v(\ell, V) \leq 2^{-j+1}$.
Output: a set of disjoint intervals-tuple pairs $(I_1, T_1), \ldots, (I_s, T_s)$ or a subset $H \subseteq B$.

  1. Let $\mathcal{I}$ be a collection of interval-tuple pairs, which is initially empty.
  2. Consider the map $q\colon B \to \{0, \ldots, \lfloor \log n \rfloor\} \cup \{\perp\}$ defined by
       $q(\ell) = \begin{cases} \perp & \forall t \in \{0,\ldots,\lfloor \log n \rfloor\}, \frac{|A_t(\ell,V)|}{2^t} < \frac{2^{-j}}{Ck\log k} \\ \max\left\{t : \frac{|A_t(\ell,V)|}{2^t} \geq \frac{2^{-j}}{Ck\log k}\right\} & \text{otherwise} \end{cases}.$
  3. Let $H = \{\ell \in B : q(\ell) = \perp\}$, and return $H$ if $|H| \geq |B|/2$.
  4. Otherwise, let $D \leftarrow B \setminus H$ and repeat the following until $D = \emptyset$:
       • Pick any $\ell \in D$ where $q(\ell) = \max_{\ell' \in D} q(\ell')$, and let $t = q(\ell)$.
       • Let $I \leftarrow [\ell - k2^{t+1}, \ell + k2^{t+1}] \cap [n]$ and $T' \leftarrow A_t(\ell, V)$.
       • Obtain $T''$ from $T'$ as follows: find a value $\nu$ such that at least $|T'|/2$ of tuples
         $(i_1, \ldots, i_{k_0}) \in T'$ satisfy $f(i_c) \leq \nu$, and at least $|T'|/2$ of tuples $(i_1, \ldots, i_{k_0}) \in T'$
         satisfy $f(i_{c+1}) > \nu$ ($\nu$ could be taken to be the median of the multiset $\{f(i_c) :$
         $(i_1, \ldots, i_{k_0}) \in T'\}$). Recombine these prefixes and suffixes (matching them in one-
         to-one correspondence) to obtain a set of disjoint $k_0$-tuples $T''$ of size $|T''| \geq |T'|/2$.
       • Append $(I, T'')$ to $\mathcal{I}$, and let $D \leftarrow D \setminus [\ell - 2 \cdot k2^{t+1}, \ell + 2 \cdot k2^{t+1}]$.
  5. return $\mathcal{I}$.
```

Figure 3.5: Description of the `GreedyDisjointIntervals` subroutine.

Proof. It is clear that the algorithm always terminates, and outputs either a collection $\mathcal{I}$ of interval-tuple pairs or a subset $H \subseteq B$. Suppose that the input of the algorithm, (f, B_{j^*}, j^*), satisfies (3.12), and consider the two possible types of outputs.

If the algorithm returns a set $H \subseteq B_{j^*}$ (in step 3), then we have $|H| \geq \frac{|B|}{2} \geq \frac{1}{2} \cdot 2^{j^*}\mu n$ (the second inequality by (3.12)). (To see why the elements of H start $(4 \cdot 2^{-j^*}/(Ck\log k), 2^{-j^*}/(12\log k))$-growing suffixes (Definition 3.6), notice that we may apply Lemma 3.16 with $\ell' = \ell$ and $\beta =$

$2^{-j^*}/(Ck\log k)$.)

If, instead, the algorithm returns a collection $\mathcal{I} = ((I_h, T_h) : h \in [s])$ in step 5, we have that, by construction, each T_h is obtained from a set $T'_h = A_t(\ell, V)$ for some ℓ with $q(\ell) \neq \bot$. Consequently, for all $h \in [s]$ we have

$$\frac{|T_h|}{|I_h|} \geq \frac{|T'_h|}{2|I_h|} \geq \frac{|A_{q(\ell)}(\ell, V)|}{4k \cdot 2^{q(\ell)+1}} \geq \frac{1}{8k} \cdot \frac{2^{-j^*}}{Ck\log k}. \tag{3.13}$$

(from the definition of $q(\ell)$). To argue that $\sum_{h=1}^{s} |I_h|$ is large, observe that, since we did not output the set H, we must have had $|D| > |B_{j^*}|/2$. Since, when adding (I_h, T_h) (corresponding to some ℓ_h) to $\mathcal{I}$ we remove at most $4k2^{q(\ell)+1} = 2|I_h|$ elements from D, in order to obtain an empty set D and reach step 5 we must have $\sum_{h=1}^{s} |I_h| \geq |B_{j^*}|/4$, which is at least $2^{j^*}\mu n/4$ by (3.12). Moreover, the sets I_h are disjoint: this is because of our choice of maximal $q(\ell)$ in step 4, which ensures that after removing $[\ell - 2k2^{q(\ell)+1}, \ell + 2k2^{q(\ell)+1}]$ in step 4 there cannot remain any $\ell' \in D$ with $[\ell' - k2^{q(\ell')+1}, \ell' + k2^{q(\ell')+1}] \cap I_h \neq \emptyset$.

Thus, it remains to prove that $\mathcal{I}$ is a $(c, 1/(6k), 2^{-j^*}/(8Ck^2\log k))$-splittable collection. To do so, consider any $(I_h, T_h) \in \mathcal{I}$. The first condition in Definition 3.7 of splittable pairs, namely that $|T_h|/|I_h| \geq 2^{-j^*}/(8Ck^2\log k)$ holds due to (3.13). Recalling step 4, we have $I_h = [\ell - k2^{t+1}, \ell + k2^{t+1}]$ for some ℓ, where $t = q(\ell)$, and T_h obtained from $T'_h = A_t(\ell, V)$. Set

$$L_h := [\ell - k2^{t+1}, \ell - \gamma 2^t], \quad M_h := (\ell - \gamma 2^t, \ell + \gamma 2^t), \quad R_h := [\ell + \gamma 2^t, \ell + k2^{t+1}].$$

This is a partition of I_h into three adjacent intervals whose size is at least $|I_h|/(6k)$ (recall that $\gamma = 1/3$). Moreover, for every $(i_1, \ldots, i_{k_0}) \in T'_h$, the c-prefix $(i_1, \ldots, i_c)$ is in $(L_h)^c$ while the $(k_0 - c)$-suffix $(i_{c+1}, \ldots, i_{k_0})$ is in $(R_h)^{k_0-c}$, by the first item of Lemma 3.14. Since T_h is obtained from a subset of these very prefixes and suffices, the conclusion holds for T_h as well. Moreover, our construction of T_h from T'_h guarantees that the last requirement in Definition 3.7 holds: for every prefix $(i_1, \ldots, i_c)$ of a tuple in T_h and suffix $(j_1, \ldots, j_{k_0-c})$ of a tuple in T_h, we have $f(i_c) < f(j_1)$. This shows that (I_h, T_h) is $(c, 1/(6k), 2^{-j^*}/(8Ck^2\log k))$-splittable, and overall that $\mathcal{I}$ is a $(c, 1/(6k), 2^{-j^*}/(8Ck^2\log k))$-splittable collection as claimed. □

Theorem 3.8 follows by executing `GreedyDisjointIntervals`(f, B_{j^*}, j^*). If the algorithm outputs a set $H \subseteq B_{j^*}$, set $\alpha = 4\cdot 2^{-j^*}/(Ck\log k)$, so we have identified a subset H of $(\alpha, C'\alpha k)$-growing suffixes (where $C' = C/48$) satisfying $\alpha|H| \geq \delta n/\operatorname{poly}(k, \log(1/\delta)) = |T_0|/\operatorname{poly}(k, \log(1/\delta))$ (using the definition of μ before (3.12)). Otherwise, set $\alpha = 2^{-j^*}/(8Ck^2\log k)$, and the algorithm outputs a $(c, 1/(6k), \alpha)$-splittable collection $\{(I_1, T_1), \ldots, (I_s, T_s)\}$ of the set $T' := \cup_{h\in[s]} T_h$. Clearly, $E(T') \subseteq E(T)$, and moreover, $\alpha \sum_{h=1}^{s} |I_h| \geq \delta n/\operatorname{poly}(k, \log(1/\delta)) = |T_0|/\operatorname{poly}(k, \log(1/\delta))$. In fact, $2^{-j^*} = \Omega(\delta/k^2)$ and so $\alpha \geq \Omega(\delta/(k^4\log k))$.

3.4 The Algorithm

3.4.1 High-level Plan

We now present the non-adaptive algorithm for finding monotone subsequences of length k.

Theorem 3.19. *Consider any fixed value of $k \in \mathbb{N}$. There exists a non-adaptive and randomized algorithm,* $\texttt{Sampler}_k(f, \varepsilon)$*, which takes two inputs: query access to a function $f\colon [n] \to \mathbb{R}$ and a parameter $\varepsilon > 0$. If f is ε-far from $(12\dots k)$-free, then* $\texttt{Sampler}_k(f, \varepsilon)$ *finds a $(12\dots k)$-pattern with probability at least $9/10$. The query complexity of* $\texttt{Sampler}_k(f, \varepsilon)$ *is at most*

$$\frac{1}{\varepsilon}\left(\frac{\log n}{\varepsilon}\right)^{\lfloor \log_2 k \rfloor} \cdot \mathrm{poly}(\log(1/\varepsilon))\,.$$

The particular dependence on k and $\log(1/\varepsilon)$ obtained from Theorem 3.19 is on the order of $(k\log(1/\varepsilon))^{O(k^2)}$. The algorithm is divided into two cases, corresponding to the two outcomes from an application of Theorem 3.11. Suppose $f\colon [n] \to \mathbb{R}$ is a function which is ε-far from being $(12\dots k)$-free. By Theorem 3.11 one of the following holds, where $C > 0$ is a large constant.

Case 1: there exist $\alpha \geq \varepsilon/\mathrm{polylog}(1/\varepsilon)$ and a set $H \subseteq [n]$ of $(\alpha, Ck\alpha)$-growing suffixes where $\alpha|H| \geq \varepsilon n/\mathrm{polylog}(1/\varepsilon)$, or

Case 2: there exist a set $T \subseteq [n]^k$ of disjoint, length-k monotone sequences, that satisfies $|T| \geq \varepsilon n/(\mathrm{polylog}(1/\varepsilon))$, and a k-tree descriptor (G, ϱ, I) which represents $(f, T, [n])$.

Theorem 3.19 follows from analyzing the two cases independently, and designing an algorithm for each.

Lemma 3.20 (Case 1). *Consider any fixed value of $k \in \mathbb{N}$, and let $C > 0$ be a large enough constant. There exists a non-adaptive and randomized algorithm,* $\texttt{Sample-Suffix}_k(f, \varepsilon)$ *which takes two inputs: query access to a function $f\colon [n] \to \mathbb{R}$ and a parameter $\varepsilon > 0$. Suppose there exist $\alpha \in (0,1)$ and a set $H \subseteq [n]$ of $(\alpha, Ck\alpha)$-growing suffixes satisfying $\alpha|H| \geq \varepsilon n/\mathrm{polylog}(1/\varepsilon)$,*[11] *then* $\texttt{Sample-Suffix}_k(f, \varepsilon)$ *finds a length-k monotone subsequence of f with probability at least $9/10$. The query complexity of* $\texttt{Sample-Suffix}_k(f, \varepsilon)$ *is at most*

$$\frac{\log n}{\varepsilon} \cdot \mathrm{polylog}(1/\varepsilon).$$

Lemma 3.20 above, which corresponds to the first case of Theorem 3.11, is proved in Section 3.4.2.

[11] Here we think of k as fixed, so $\mathrm{polylog}(1/\varepsilon)$ is allowed to depend on k. In this lemma, the expression stands for $(k\log(1/\varepsilon))^k$.

Lemma 3.21 (Case 2)**.** *Consider any fixed value of $k \in \mathbb{N}$. There exists a non-adaptive, randomized algorithm,* `Sample-Splittable`$_k(f, \varepsilon)$ *which takes two inputs: query access to a sequence $f\colon [n] \to \mathbb{R}$ and a parameter $\varepsilon > 0$. Suppose there exists a set $T \subseteq [n]^k$ of disjoint, length-k monotone subsequences of f where $|T| \geq \varepsilon n/\mathrm{polylog}(1/\varepsilon)$,*[12] *as well as a (k, k, α)-tree descriptor (G, ϱ, I) that represents $(f, T, [n])$, where $\alpha \geq \varepsilon/\mathrm{polylog}(1/\varepsilon)$, then* `Sample-Splittable`$_k(f, \varepsilon)$ *finds a length-k monotone subsequence of f with probability at least $9/10$. The query complexity of* `Sample-Splittable`$_k(f, \varepsilon)$ *is at most*

$$\frac{1}{\varepsilon}\left(\frac{\log n}{\varepsilon}\right)^{\lfloor \log_2 k \rfloor} \cdot \mathrm{polylog}(1/\varepsilon).$$

Proof of Theorem 3.19 assuming Lemmas 3.20 and 3.21. The algorithm `Sampler`$_k(f, \varepsilon)$ executes both `Sample-Suffix`$_k(f, \varepsilon)$ and `Sample-Splittable`$_k(f, \varepsilon)$; if either algorithm finds a length-k monotone subsequence of f, output such a subsequence. We note that by Theorem 3.11, either case 1, or case 2 holds. If case 1 holds, then by Lemma 3.20, `Sample-Suffix`(f, ε) outputs a length-k monotone subsequence with probability at least $9/10$, and if case 2 holds, then by Lemma 3.21, `Sample-Splittable`$_k(f, \varepsilon)$ outputs a length-k monotone subsequence with probability at least $9/10$. Thus, regardless of which case holds, a length-k monotone subsequence will be found with probability at least $9/10$. The query complexity then follows from the maximum of the two query complexities. □

3.4.2 Proof of Lemma 3.20: An Algorithm for Growing Suffixes

We now prove Lemma 3.20. Let $C > 0$ be a large constant, and let $k \in \mathbb{N}$ be fixed. Let $\varepsilon > 0$ and $f\colon [n] \to \mathbb{R}$ be a function which is ε-far from $(12 \ldots k)$-free. Furthermore, as per the assumption of case 1 of the algorithm, we assume that there exists a parameter $\alpha \in (0, 1)$ as well as a set $H \subseteq [n]$ of $(\alpha, Ck\alpha)$-growing suffixes, where $\alpha|H| \geq \varepsilon n/\mathrm{polylog}(1/\varepsilon)$.

The algorithm, which underlies the result of Lemma 3.20, proceeds by sampling uniformly at random an index $\boldsymbol{a} \sim [n]$, and running a sub-routine which we call `Growing-Suffix`, with $\boldsymbol{a}$ as input. The sub-routine is designed so that if $\boldsymbol{a}$ is the start of an $(\alpha, Ck\alpha)$-growing suffix then the algorithm will find a length-k monotone subsequence of f with probability at least $99/100$. The sub-routine, `Growing-Suffix`, is presented in Figure 3.6.

Lemma 3.22. *Let $f\colon [n] \to \mathbb{R}$ be a function, let $\alpha, \alpha_0, \beta \in (0, 1)$ be parameters satisfying $\beta \geq Ck\alpha$ and $\alpha_0 \leq \alpha$, and suppose that $a \in [n]$ starts a (α, β)-growing suffix in f. Then the procedure* `Growing-Suffix`(f, α_0, a) *finds a $(12 \ldots k)$-copy in f with probability at least $99/100$.*

Proof. Recall, from Definition 3.6, that if $a \in [n]$ is the start of a (α, β)-growing suffix of f then there exist a collection of sets, $D_1(a), \ldots, D_{\eta_a}(a)$ and parameters $\delta_1(a), \ldots, \delta_{\eta_a}(a) \in (0, \alpha]$, where

[12]in this case the polylog$(1/\varepsilon)$ term stands for $(k \log(1/\varepsilon))^{O(k^2)}$

Subroutine `Growing-Suffix`(f, α_0, a)

Input: Query access to a function $f\colon [n] \to \mathbb{R}$, a parameter $\alpha_0 \in (0,1)$, and an index $a \in [n]$.
Output: a subset of k indices $i_1 < \cdots < i_k$ where $f(i_1) < \cdots < f(i_k)$, or fail.

1. Let $\eta_a = \lceil \log(n-a) \rceil$ and consider the sets $S_j(a) = (a+\ell_{j-1}, a+\ell_j] \cap [n]$ for all $j \in [\eta_a]$ and $\ell_j = 2^j$.
2. For each $j \in [\eta_a]$, let $\mathbf{A}_j \subseteq S_j(a)$ be obtained by sampling uniformly at random $T := 1/\alpha_0$ times from $S_j(a)$.
3. For each $j \in [\eta_a]$ and each $b \in \mathbf{A}_j$, query $f(b)$.
4. If there exist indices $i_1, \ldots, i_k \in \mathbf{A}_1 \cup \cdots \cup \mathbf{A}_{\eta_i}$ satisfying $i_1 < \cdots < i_k$ and $f(i_1) < \cdots < f(i_k)$, **return** such indices $i_1, \ldots, i_k$. Otherwise, **return** fail.

Figure 3.6: Description of the `Growing-Suffix` subroutine.

every $j \in [\eta_a]$ has

$$D_j(a) \subseteq S_j(a), \qquad |D_j(a)| = \delta_j(a) \cdot |S_j(a)|, \qquad \text{and} \qquad \sum_{j=1}^{\eta_a} \delta_j(a) \geq \beta.$$

Further, if, for some $j_1, \ldots, j_k \in [\eta_i]$, we have $j_1 < \cdots < j_k$ and for all $\ell \in [k]$, $\mathbf{A}_{j_\ell} \cap D_{j_\ell}(a) \neq \emptyset$, then the union $D_{j_1}(a) \cup \ldots \cup D_{j_k}(a)$ contains a length-k monotone subsequence. In view of this, for each $j \in [\eta_a]$, consider the indicator random variable

$$\mathbf{E}_j := \operatorname{ind}\{\mathbf{A}_j \cap D_j(a) \neq \emptyset\},$$

and observe that by the foregoing discussion `Growing-Suffix`(f, α_0, a) samples a length-k monotone subsequence of f whenever $\sum_{j=1}^{\eta_a} \mathbf{E}_j \geq k$. We note that the $\mathbf{E}_j$'s are independent, and that

$$\mathbf{Pr}[\mathbf{E}_j = 1] = 1 - (1-\delta_j(a))^T \geq \min\left\{\frac{T \cdot \delta_j(a)}{10}, \frac{1}{10}\right\}.$$

Let $J \subseteq [\eta_a]$ be the set of indices satisfying $T \cdot \delta_j(a) \geq 1$ (recall that $T = 1/\alpha_0$). Then, if $|J| \geq Ck$ we have

$$\mathbb{E}\left[\sum_{j=1}^{\eta_a} \mathbf{E}_j\right] \geq \frac{Ck}{10},$$

since every variable $j \in J$ contributes at least $1/10$. On the other hand, if $|J| \leq Ck/2$, then, since $\delta_j(a) \leq \alpha$ for every j, we have $\sum_{j \in [\eta_a] \setminus J} \delta_j(a) \geq \beta - |J| \cdot \alpha \geq \beta/2$ (using $\beta \geq Ck\alpha$) so that

$$\mathbb{E}\left[\sum_{j=1}^{\eta_a} \mathbf{E}_j\right] \geq \mathbb{E}\left[\sum_{j \in [\eta_a] \setminus J} \mathbf{E}_j\right] \geq \frac{T}{10} \cdot \frac{\beta}{2} \geq \frac{Ck}{20}.$$

In either case, $\mathbb{E}[\sum_{j\in[\eta_a]}\mathbf{E}_j] \geq Ck/20$, and since the events $\mathbf{E}_i$ are independent, via a Chernoff bound we obtain that $\sum_j \mathbf{E}_j$ is larger than k with probability at least 99/100. □

```
Subroutine Sample-Suffix_k (f, ε)

Input: Query access to a function $f\colon [n] \to \mathbb{R}$, and a parameter $\varepsilon \in (0,1)$.
Output: a subset of $k$ indices $i_1 < \cdots < i_k$ where $f(i_1) < \cdots < f(i_k)$, or fail.

  1. Repeat the following for all $j = 1, \dots, O(\log(1/\varepsilon))$, letting $\alpha_j = 2^{-j}$:
       • For $t_j = \alpha_j \cdot \mathrm{polylog}(1/\varepsilon)/\varepsilon$ iterations, sample $\boldsymbol{a} \sim [n]$ uniformly at random and
         run Growing-Suffix($f, \alpha_j, \boldsymbol{a}$), and if it returns a length-$k$ monotone subsequence
         of $f$, return that subsequence.
  2. If the algorithm has not already output a monotone subsequence, return fail.
```

Figure 3.7: Description of the `Sample-Suffix` subroutine.

With this in hand, we can now establish Lemma 3.20.

Proof of Lemma 3.20. First, note that the query complexity of `Sample-Suffix`$_k(f,\varepsilon)$ is

$$\sum_{j=1}^{O(\log(1/\varepsilon))} t_j \cdot O(\log n/\alpha_j) = \frac{\log n \cdot \mathrm{polylog}(1/\varepsilon)}{\varepsilon}.$$

Consider the iteration of j where $\alpha_j \leq \alpha \leq 2\alpha_j$ (note that since $\alpha \geq \varepsilon/\mathrm{polylog}(1/\varepsilon)$, there exists such j). Then, since $|H| \geq \varepsilon/(\alpha\cdot\mathrm{polylog}(1/\varepsilon))$, we have that $t_j \geq Cn/|H|$ (for a sufficiently large constant C). Thus, with probability at least 99/100, some iteration satisfies $\boldsymbol{a} \in H$. When this occurs, `Growing-Suffix`$(f, \alpha_j, \boldsymbol{a})$ will output a length-k monotone subsequence with probability at least 99/100, by Lemma 3.22, and thus by a union bound we obtain the desired result. □

3.4.3 Proof of Lemma 3.21: An Algorithm for Splittable Intervals

We now prove Lemma 3.21. We consider a fixed setting of $k \in \mathbb{N}$ and $\varepsilon > 0$, and let $f\colon [n] \to \mathbb{R}$ be any sequence which is ε-far from being $(12\dots k)$-free. Furthermore, as per case 2 of the algorithm, we assume that there exists a set $T \subseteq [n]^k$ of disjoint length-k monotone subsequences of f where

$$|T| \geq \frac{\varepsilon n}{\mathrm{polylog}(1/\varepsilon)},$$

and (G, ϱ, I) is a (k, k, α)-tree descriptor which represents $(f, T, [n])$, where $\alpha \geq \varepsilon/\mathrm{polylog}(1/\varepsilon)$. In what follows, we describe a sub-routine, `Sample-Splittable`$_k(f,\varepsilon)$ in terms of two parameters

$\rho, q \in \mathbb{R}$. The parameter $\rho > 0$ is set to be sufficiently large and independent of n, satisfying

$$\rho \geq \frac{\varepsilon}{\mathrm{polylog}(1/\varepsilon)}. \tag{3.14}$$

One property which we will want to satisfy is that if we take a random subset of $[n]$ by including each element independently with probability $1/(\rho n)$, we will include an element belonging to $E(T)$ with probability at least $1 - 1/(Ck)$, for a large constant $C > 0$. The parameter q will be an upper bound on the query complexity of the algorithm, which we set to a high enough value satisfying:

$$q = O\left(\frac{1}{\rho}\left(\frac{\log n}{\rho}\right)^{\lfloor \log_2 k \rfloor}\right) \leq \frac{1}{\varepsilon} \cdot \left(\frac{\log n}{\varepsilon}\right)^{\lfloor \log_2 k \rfloor} \cdot \mathrm{polylog}(1/\varepsilon).$$

```
Subroutine Sample-Splittable_k (f, ε)

Input: Query access to a sequence $f\colon [n] \to \mathbb{R}$, and a parameter $\varepsilon \in (0,1)$.
Output: a subset of $k$ indices $i_1 < \cdots < i_k$ where $f(i_1) < \cdots < f(i_k)$, or fail.

  1. Let $r = \lfloor \log_2 k \rfloor$ and run Sample-Helper($r, [n], \rho$), to obtain a set $\mathbf{A} \subseteq [n]$.
  2. If $|\mathbf{A}| > q$, return fail; otherwise, for each $a \in \mathbf{A}$, query $f(a)$. If there exists a monotone
     sequence of $f$ of length $k$, then return that subsequence. If not, return fail.
```

Figure 3.8: Description of the `Sample-Splittable` subroutine.

The descriptions of the main algorithm `Sample-Splittable`$_k$ and the sub-routine `Sample-Helper`, are given in Figure 3.8 and Figure 3.9. Note that, for any $r \in \mathbb{N}$, if we let $\mathcal{D}_r$ be the distribution of $|\mathbf{A}|$, where $\mathbf{A}$ is the output of a call to `Sample-Helper`$(r, [n], \rho)$. Then, we have that $\mathcal{D}_0 = \mathrm{Bin}(n, \rho)$, and for $r > 0$, $\mathcal{D}_r$ is stochastically dominated by the random variable

$$\sum_{i=1}^{\boldsymbol{y}_0} \sum_{j=1}^{O(\log n)} \boldsymbol{x}_{r-1}^{(i,j)},$$

where $\boldsymbol{y}_0 \sim \mathrm{Bin}(n, 1/(\rho n))$ and $\boldsymbol{x}_{r-1}^{(i,j)} \sim \mathcal{D}_{r-1}$ for all $i \in \mathbb{N}$ and $j \in [O(\log n)]$ are all mutually independent. As a result, for $r \geq 1$,

$$\mathbb{E}\left[|\mathbf{A}|\right] \leq \frac{1}{\rho} \cdot \log n \cdot \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}_{r-1}}[\boldsymbol{x}],$$

and since $\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_0}[\boldsymbol{x}] = 1/\rho$, we have:

$$\mathbb{E}\left[|\mathbf{A}|\right] \leq \frac{1}{\rho}\left(\frac{\log n}{\rho}\right)^r.$$

```
Subroutine Sample-Helper$(r, I, \rho)$

Input: An integer $r \in \mathbb{N}$, an interval $I \subseteq [n]$, and a parameter $\rho \in (0,1)$.
Output: a subset of $A \subseteq I$.

1. Let $\mathbf{A}_0 = \emptyset$. For every index $a \in I$, let $\mathbf{A}_0 \leftarrow \mathbf{A}_0 \cup \{a\}$ with probability $1/(\rho|I|)$.
2. If $r = 0$, return $\mathbf{A}_0$.
3. If $r > 0$, proceed with the following:
    • For every index $a \in \mathbf{A}_0$, consider the $O(\log n)$ intervals given by $B_{a,j} = [a - \ell_j, a + \ell_j]$, for $j = 1, \dots, O(\log n)$ and $\ell_j = 2^j$, and let $\mathbf{R}_{a,j} \leftarrow$ Sample-Helper$(r - 1, B_{a,j}, \rho)$.
    • Let $\mathbf{A}$ be the set
          $\mathbf{A} \leftarrow \bigcup_{a \in \mathbf{A}_0,\, j = O(\log n)} \mathbf{R}_{a,j}$.
    • return the set $(\mathbf{A}_0 \cup \mathbf{A}) \cap I$.
```

Figure 3.9: Description of the `Sample-Helper` subroutine.

We may then apply Markov's inequality to conclude that $|\mathbf{A}| \leq q$ with probability at least 99/100. As a result, we focus on proving that the probability that the set $\mathbf{A}$ contains a monotone subsequence of f of length k is at least 99/100. This would imply the desired result by taking a union bound.

In addition to the above, we define another algorithm, `Sample-Helper`*, in Figure 3.10, which will be a *helper* sub-routine. We emphasize that `Sample-Helper`* is not executed in the algorithm itself, but will be useful in order to analyze `Sample-Helper`.

```
Subroutine Sample-Helper$^*(r, I, \rho, \mathcal{I})$

Input: An integer $r \in \mathbb{N}$, an interval $I \subseteq [n]$, a parameter $\rho \in (0,1)$, and a collection of
disjoint intervals $\mathcal{I}$ of $[n]$.
Output: two subsets $\mathbf{A}, \mathbf{A}_0 \subseteq I$.

  1. Let $\mathbf{A}_0 = \emptyset$. For every index $a \in I$ which lies inside an interval in $\mathcal{I}$, let $\mathbf{A}_0 \leftarrow \mathbf{A}_0 \cup \{a\}$
     with probability $1/(\rho|I|)$.

  2. If $r = 0$, return $\mathbf{A}_0$.

  3. If $r > 0$, proceed with the following:

     • For every index $a \in \mathbf{A}_0$, consider the $O(\log n)$ intervals given by $B_{a,j} =
       [a - \ell_j, a + \ell_j]$, for $j = 1, \ldots O(\log n)$, and $\ell_j = 2^j$, and let $(\mathbf{R}_{a,j}, \mathbf{R}_{a,j,0}) \leftarrow$
       Sample-Helper$^*(r-1, B_{a,j}, \rho, \mathcal{I})$.

     • Let $\mathbf{A}$ to be the set
           $\mathbf{A} \leftarrow \bigcup_{a \in \mathbf{A}_0,\ j = O(\log n)} \mathbf{R}_{a,j}.$

     • return the set $(\mathbf{A} \cap I, \mathbf{A}_0 \cap I)$.
```

Figure 3.10: Description of the `Sample-Helper`* subroutine.

Before proceeding, we require a "coupling lemma." Its main purpose is to prove the intuitive fact that if $\mathcal{I}_0, \mathcal{I}_1$ are collections of disjoint intervals, and the latter is a refinement of the former (namely, each intervals in $\mathcal{I}_1$ is contained in an interval of $\mathcal{I}_0$), then `Sample-Helper`$^*(r, [n], \rho, \mathcal{I}_0)$ is more likely to find a length-k monotone subsequence than `Sample-Helper`$^*(r, [n], \rho, \mathcal{I}_1)$ does.

Lemma 3.23. *Let $r \in \mathbb{N}$ be an integer, $f\colon [n] \to \mathbb{R}$ a function, $\rho \in (0,1)$ a parameter, and $\mathcal{I}_0$ and $\mathcal{I}_1$ collections of disjoint intervals in $[n]$, such that each interval in $\mathcal{I}_1$ lies inside an interval from $\mathcal{I}_0$. Denote by $(\mathbf{A}^{(i)}, \mathbf{A}_0^{(i)})$ the random pair of sets given by the output of* `Sample-Helper`$^*(r, [n], \rho, \mathcal{I}_i)$*, for $i = 0, 1$. Lastly, let $\mathcal{E}\colon \mathcal{P}([n]) \times \mathcal{P}([n]) \to \{0,1\}$ be any* monotone *function; that is, it satisfies $\mathcal{E}(S_1, S_2) \leq \mathcal{E}(S_1', S_2')$ for any $S_1 \subseteq S_1' \subseteq [n]$ and $S_2 \subseteq S_2' \subseteq [n]$. Then,*

$$\mathbf{Pr}[\mathcal{E}(\mathbf{A}^{(0)}, \mathbf{A}_0^{(0)}) = 1] \geq \mathbf{Pr}[\mathcal{E}(\mathbf{A}^{(1)}, \mathbf{A}_0^{(1)}) = 1].$$

Proof. Consider an execution of `Sample-Helper`$^*(r, [n], \rho, \mathcal{I}_0)$ which outputs a pair $(\mathbf{A}^{(0)}, \mathbf{A}_0^{(0)})$. Let $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(1)}$ be the subsets of $\mathbf{A}^{(0)}$ and $\mathbf{A}^{(0)}$, respectively, obtained by running a parallel execution of `Sample-Helper`$^*(r, [n], \rho, \mathcal{I}_1)$, which follows the execution of `Sample-Helper`$^*(r, [n], \rho, \mathcal{I}_0)$, but whenever an element which is not in an interval of $\mathcal{I}_1$ is considered, it is simply ignored (i.e., it is not included in $\mathbf{A}^{(0)}$ or in $\mathbf{A}_0^{(0)}$ and no recursive calls based on such elements are made). It is easy to see that this coupling yields a pair $(\mathbf{A}^{(1)}, \mathbf{A}_0^{(1)})$ with the same distribution as that given by running `Sample-Helper`$^*(r, [n], \rho, \mathcal{I}_1)$. As $\mathcal{E}(\cdot,\cdot)$ is increasing, if $\mathcal{E}(\mathbf{A}^{(0)}, \mathbf{A}_0^{(0)})$ holds then so does $\mathcal{E}(\mathbf{A}^{(1)}, \mathbf{A}_0^{(1)})$. The lemma follows. $\square$

The following corollary is a direct consequence of Lemma 3.23. Specifically, we use the facts that `Sample-Splittable`$_k(f,\varepsilon)$ calls `Sample-Helper`$(\lfloor \log_2 k \rfloor, [n], \rho)$, which is equivalent to calling `Sample-Helper`$(\lfloor \log_2 k \rfloor, [n], \rho, \{[n]\})$, and that finding a $(12\ldots k)$-pattern in $\mathcal{I}$ is a monotone event.

Corollary 3.24. *Let $\mathcal{I}$ be any collection of disjoint intervals in $[n]$. Suppose $(\mathbf{A}, \mathbf{A}_0)$ is the random pair of sets given by the output of* `Sample-Helper`$^*(\lfloor \log_2 k \rfloor, n, \rho, \mathcal{I})$*, then,*

$$\begin{aligned}\mathbf{Pr}[\texttt{Sample-Splittable}_k(f,\varepsilon) \text{ finds a } (12\ldots k)\text{-pattern of } f] \geq \\ \mathbf{Pr}[\mathbf{A} \text{ contains a } (12\ldots k)\text{-pattern in } f_{|\mathcal{I}}].\end{aligned}$$

Definition 3.25. *Let $k_0 \in \mathbb{N}$ be a positive integer, and let (G, ϱ) be a k_0-tree descriptor (for this definition we do not care about the third component of the descriptor, I). We say that $p \in [k_0]$ is the* primary index *of (G,ϱ) if the leaf with rank p under $\leq_G$ is the unique leaf whose root-to-leaf path $(u_1, \ldots, u_d)$ satisfies the following: for each $d' \in [d-1]$, denoting the left and right children of $u_{d'}$ by v_l and v_r, respectively, $u_{d'+1}$ is v_l if the number of leaves in the subtree rooted at v_l is at least the number of leaves in the subtree rooted at v_r, and otherwise, $u_{d'+1}$ is v_r.*

With Corollary 3.24 in hand, we note that Lemma 3.21 follows from the next lemma.

Lemma 3.26. *Let $k, k_0, n \in \mathbb{N}$ satisfy $1 \leq k_0 \leq k$, let C be a large enough constant, and let $\alpha, \rho \in (0,1)$ be such that $\rho \geq C\alpha$ and $\alpha \geq \rho/\mathrm{polylog}(1/\rho)$. Let $f\colon [n] \to \mathbb{R}$ be a function, let $\mathcal{I}$ be a collection of disjoint intervals in $[n]$, for each $I \in \mathcal{I}$ let $T_I \subseteq I^{k_0}$ be a set of disjoint, length-k_0 monotone subsequence of f, and suppose that*

$$\sum_{I \in \mathcal{I}} |T_I| \geq \alpha n/4.$$

Suppose that (G, ϱ) is a (k, k_0, α)-weighted-tree such that for every $I \in \mathcal{I}$ there exists a function $\mathsf{I}_I : V(G) \to \mathcal{S}(I)$, such that $(G, \varrho, \mathsf{I}_I)$ is a tree descriptor that represents (f, T_I, I). Given any $r \in \mathbb{N}$ satisfying $\lfloor \log_2 k_0 \rfloor \leq r$, let $(\mathbf{A}, \mathbf{A}_0)$ be the pair of sets output by the sub-routine `Sample-Helper`$^*(r, [n], \rho, \mathcal{I})$*. With probability at least $1 - k_0/(100k)$, there exist indices $i_1, \ldots, i_{k_0} \in [n]$ with the following properties.*

1. *$(i_1, \ldots, i_{k_0})$ is a length-k_0 monotone subsequence of f.*
2. *There is an interval $I \in \mathcal{I}$ such that $i_1, \ldots, i_{k_0} \in I \cap E(T_I)$.*
3. *$i_1, \ldots, i_{k_0} \in \mathbf{A}$ and $i_p \in \mathbf{A}_0$, where p is the primary index of (G, ϱ).*

Proof. The proof proceeds by induction on k_0. Consider the base case, when $k_0 = 1$. In this case, $\lfloor \log_2 k_0 \rfloor = 0$, so for any $r \geq 0$, $\texttt{Sample-Helper}^*(r, [n], \rho, \mathcal{I})$ runs step 1. As a result, $\texttt{Sample-Helper}^*(r, [n], \rho, \mathcal{I})$ samples each element inside an interval of $\mathcal{I}$ independently with probability $1/(\rho n)$. In order to satisfy the requirements of the lemma in this case, we need $\mathbf{A}_0$ to contain an element of $\cup_{I\in\mathcal{I}} T_I$. By the assumption on the size of this union, and because each of the elements of the union lives inside some interval from $\mathcal{I}$, such an element will exist with sufficiently high probability via a Chernoff bound.

For the inductive step, assume that Lemma 3.26 is fulfilled whenever $k_0 < K$, for $K \in \mathbb{N}$ satisfying $1 < K \leq k$, and we will prove, assuming this inductive hypothesis, that Lemma 3.26 holds for $k_0 = K$. So consider a setting $k_0 = K$. Let $\mathcal{I}$, (G, ϱ) and I_I be as in the statement of the lemma. Denote the root of (G, ϱ) by v_{root}, and its left and right children by v_{left} and v_{right}. Let c be the number of leaves in the subtree $(G_{\mathsf{left}}, \varrho_{\mathsf{left}})$ rooted at v_{left}, so $k_0 - c$ is the number of leaves in the subtree $(G_{\mathsf{right}}, \varrho_{\mathsf{right}})$ rooted at v_{right}. We shall assume that $c \geq k_0 - c$; the other case follows by an analogous argument.

For each $I \in \mathcal{I}$, the collection of pairs $(J, T_{I,J})$, where $J \in \mathsf{I}_I(v_{\mathsf{root}})$ and $T_J = T_I \cap J^{k_0}$ is the restriction of T_I to J, is a $(c, 1/(6k), \alpha)$-splittable collection of I. Let $\mathcal{J}$ be the collection of all such intervals J (note that they are pairwise disjoint and that $\mathcal{J}$ is a refinement of $\mathcal{I}$). Let (L_J, M_J, R_J) be the partition of J into left, middle and right intervals, respectively, and let $T_J^{(L)}$ and $T_J^{(R)}$ be sets of c-prefixes and $(k_0 - c)$-suffixes of k_0-tuples from $T_{I,J}$, as given by Definition 3.7. Set

$$\mathcal{L} = \{L_J : J \in \mathcal{J}\}, \qquad \mathcal{R} = \{R_J : J \in \mathcal{J}\}, \qquad T^{(L)} = \bigcup_{J\in\mathcal{J}} T_J^{(L)}, \qquad T^{(R)} = \bigcup_{J\in\mathcal{J}} T_J^{(R)}.$$

Note that $(G_{\mathsf{left}}, \varrho_{\mathsf{left}}, \mathsf{I}_{J,\mathsf{left}})$ is a (k, c, α)-tree descriptor for (f, T_J, J), with appropriate $\mathsf{I}_{J,\mathsf{left}}$. Similarly, $(G_{\mathsf{right}}, \varrho_{\mathsf{right}}, \mathsf{I}_{J,\mathsf{right}})$ is a $(k, k_0 - c, \alpha)$-tree descriptor for (f, T_J, J), with appropriate $\mathsf{I}_{J,\mathsf{right}}$.

We consider an execution of $\texttt{Sample-Helper}^*(r, [n], \rho, \mathcal{I})$ which outputs a random pair of sets $(\mathbf{A}, \mathbf{A}_0)$. Let $\mathbf{A}^{(L)}$ and $\mathbf{A}_0^{(L)}$ be the subsets of $\mathbf{A}$ and $\mathbf{A}_0$, respectively, obtained by running a parallel execution of $\texttt{Sample-Helper}^*(r, [n], \rho, \mathcal{L})$, where, as in the proof of Lemma 3.23, we follow the execution of $\texttt{Sample-Helper}^*(r, [n], \rho, \mathcal{I})$, but whenever an element which is not in $\mathcal{L}$ is considered, we ignore it. As stated above, this coupling yields a pair $(\mathbf{A}^{(L)}, \mathbf{A}_0^{(L)})$ with the distribution given by running $\texttt{Sample-Helper}^*(r, [n], \rho, \mathcal{L})$.

For $a \in \mathbf{A}_0^{(L)}$, and any $j \in [O(\log n)]$, let $(\mathbf{A}^{(a,j)}, \mathbf{A}_0^{(a,j)})$ be the output of the recursive call (inside the execution of $\texttt{Sample-Helper}^*(r, [n], \rho, \mathcal{I})$) of $\texttt{Sample-Helper}^*(r - 1, B_{a,j}, \rho, \mathcal{R})$.

We define the collection:

$$\mathcal{S} = \left\{ (S_0, S) : \begin{array}{l} S_0 \subseteq S \subseteq E(T^{(L)}) \\ \text{there exist } i_1, \ldots, i_c \in S \text{ forming a } (12\ldots c)\text{-pattern such that } i_p \in S_0 \\ \text{there exist } J \in \mathcal{J} \text{ such that } i_1, \ldots, i_c \in L_J \end{array} \right\}.$$

For each $(S_0, S) \in \mathcal{S}$, we let $\mathsf{a}(S_0, S) \in E(T^{(L)})$ be some $i_p \in S$ such that there exist $c - 1$ indices $i_1, \ldots, i_{p-1}, i_{p+1}, i_c$, such that $(i_1, \ldots, i_c)$ forms a $(12\ldots c)$-pattern in S, and $i_1, \ldots, i_p \in L_J$ for some $J \in \mathcal{J}$. Let $\texttt{seg}(S_0, S)$ be this interval J, and let $\texttt{len}(S_0, S) \in [O(\log n)]$ be the smallest j for which $R_J \subseteq B_{a,j}$, where $a = \mathsf{a}(S_0, S)$.

Let $\mathbf{E}_L$ be the event that

$$\left(\mathbf{A}^{(L)} \cap E(T^{(L)}), \mathbf{A}_0^{(L)} \cap E(T^{(L)})\right) \in \mathcal{S},$$

and let $\mathbf{E}_L(S_0, S)$ be the event that

$$\mathbf{A}^{(L)} \cap E(T^{(L)}) = S_0 \qquad\qquad \mathbf{A}_0^{(L)} \cap E(T^{(L)}) = S,$$

so $\mathbf{E}_L = \cup_{(S_0,S)\in\mathcal{S}} \mathbf{E}_L(S_0, S)$, and the events $\mathbf{E}_L(S_0, S)$ are pairwise disjoint.

By the induction hypothesis, applied with the family $\{L_J : J \in \mathcal{J}\}$ and the corresponding sets $T_J^{(L)}$ (using $\sum_{J\in\mathcal{J}} |T_J^{(L)}| = \sum_{J\in\mathcal{J}} |T_J| \geq \alpha n/4$), we have

$$\mathbf{Pr}[\mathbf{E}_L] \geq 1 - c/(100k).$$

Let $\mathbf{E}_R(a, j)$ be the event that $a \in \mathbf{A}_0$, and in the recursive run of $\texttt{Sample-Helper}^*(r-1, B_{a,j}, \rho, \mathcal{R})$ inside $\texttt{Sample-Helper}^*(r, [n], \rho, \mathcal{I})$, there exist indices $i'_1, \ldots, i'_{k_0-c}$ such that

- $(i'_1, \ldots, i'_{k_0-c})$ form a length $(k_0 - c)$-monotone subsequence.
- $i'_1, \ldots, i'_{k_0-c} \in E(T_J^{(R)})$, where J is the interval in $\mathcal{J}$ with $i \in J$.
- $i'_1, \ldots, i'_{k_0-c} \in \mathbf{A}^{(a,j)}$ and $i'_q \in \mathbf{A}_0^{(a,j)}$, where q is the primary index of $(G_{\mathsf{right}}, \varrho_{\mathsf{right}})$.

Let $\mathbf{F}_R(a, j)$ be the event that in a run of $\texttt{Sample-Helper}^*(r-1, B_{a,j}, \rho, \mathcal{R})$, there exist $i'_1, \ldots, i'_{k_0-c}$ as above. Fix some $(S_0, S) \in \mathcal{S}$, and let $a = \mathsf{a}(S_0, S)$, $J = \texttt{seg}(S_0, S)$ and $j = \texttt{len}(S_0, S)$. We claim that

$$\mathbf{Pr}[\mathbf{E}_R(a, j) \mid \mathbf{E}_L(S_0, S)] = \mathbf{Pr}[\mathbf{F}_R(a, j)].$$

Indeed, by conditioning on $\mathbf{E}_L(S_0, S)$ we know that $a \in \mathbf{A}_0$, so there will be a recursive run of $\texttt{Sample-Helper}^*(r-1, B_{a,j}, \rho, \mathcal{R})$, and moreover the event $\mathbf{E}_L(S_0, S)$ will have no influence on the outcomes of this run.

Note that $|T_J^{(R)}| \geq \alpha|R_J| \geq \alpha|B_{a,J}|/4$. By the induction hypothesis, applied with the interval $B_{a,J}$ in place of $[n]$, the family $\{R_J\}$ and the corresponding set $T_J^{(R)}$, and the tree $(G_{\mathsf{right}}, \varrho_{\mathsf{right}})$, we find that $\mathbf{Pr}[\mathbf{F}_R(a, j)] \geq 1 - (k_0 - c)/(100k)$. We note that if both $\mathbf{E}_L(S_0, S)$ and $\mathbf{E}_R(a, j)$ hold, then there are indices $i_1, \ldots, i_c, i'_1, \ldots, i'_{k_0-c}$ such that

- $(i_1, \ldots, i_c)$ is a length-c monotone subsequence in $E(T_J^{(L)})$, and $(i'_1, \ldots, i'_{k_0-c})$ is a length-c monotone subsequence in $E(T_J^{(R)})$. In particular, $(i_1, \ldots, i_c, i'_1, \ldots, i'_{k_0-c})$ is a length-k_0 monotone subsequence that lies in $E(T_j)$.
- $i_1, \ldots, i_c, i'_1, \ldots, i'_{k_0-c} \in \mathbf{A}$ and $i_p \in \mathbf{A}_0$ (recall that p is the primary index of both G and G_{left}).

I.e. if these two events hold, then the requirements ot the lemma are satisfied. It follows that the requirements ot the lemma are satisfied with at least the following probability, using the fact that the events $\mathbf{E}_L(S_0, S)$ are disjoint.

$$
\begin{aligned}
&\sum_{(S_0,S)\in\mathcal{S}} \mathbf{Pr}[\mathbf{E}_R(\mathsf{a}(S_0,S), \mathtt{len}(S_0,S)) \text{ and } \mathbf{E}_L(S_0,S)] \\
&= \sum_{(S_0,S)\in\mathcal{S}} \mathbf{Pr}[\mathbf{E}_R(\mathsf{a}(S_0,S), \mathtt{len}(S_0,S)) \mid \mathbf{E}_L(S_0,S)] \times \mathbf{Pr}[\mathbf{E}_L(S_0,S)] \\
&\geq \sum_{(S_0,S)\in\mathcal{S}} \mathbf{Pr}[\mathbf{F}_R(\mathsf{a}(S_0,S), \mathtt{len}(S_0,S))] \times \mathbf{Pr}[\mathbf{E}_L(S_0,S)] \\
&\geq \left(1 - \frac{k_0 - c}{100k}\right) \cdot \sum_{(S_0,S)\in\mathcal{S}} \mathbf{Pr}[\mathbf{E}_L(S_0,S)] \\
&\geq \left(1 - \frac{k_0 - c}{100k}\right) \cdot \mathbf{Pr}[\mathbf{E}_L] \\
&\geq \left(1 - \frac{k_0 - c}{100k}\right) \cdot \left(1 - \frac{c}{100k}\right) \geq 1 - \frac{k_0}{100k}.
\end{aligned}
$$

This completes the proof of Lemma 3.26. □

Chapter 4

Monotone Patterns: An Adaptive $O(\log n)$ Algorithm

The results in this chapter appear in [27].

4.1 Introduction

In this chapter we continue the investigation of testing for monotone patterns, as presented in the previous chapter. The main result is an *adaptive* algorithm with optimal dependence in n for solving the above problem. In contrast, the result presented before were for non-adaptive algorithms.

Theorem 4.1. *Fix $k \in \mathbb{N}$. For any $\varepsilon > 0$, there exists an algorithm that, given query access to a function $f\colon [n] \to \mathbb{R}$ which is ε-far from $(12 \dots k)$-free, outputs a length-k monotone subsequence of f with probability at least $9/10$, with query complexity and running time of $O_{k,\varepsilon}(\log n)$.*

For the precise bound on the query complexity and running time, see Lemma 4.8. Note that the algorithm underlying Theorem 4.1 solves the testing problem with *one-sided error*, since a length-k monotone subsequence is evidence for not being $(12 \dots k)$-free. The algorithm improves upon the non-adaptive query complexity $O_{k,\varepsilon}((\log n)^{\lfloor \log_2 k \rfloor})$ discussed in the previous chapter, and in particular, breaks the non-adaptive lower bound [22]. Hence, Theorem 4.1 implies a natural separation between the power of adaptive and non-adaptive algorithms for finding monotone subsequences.

Theorem 4.1 is optimal, even among two-sided error algorithms. In the case $k = 2$, corresponding to monotonicity testing, there is a $\Omega(\log n/\varepsilon)$ lower bound (as long as, say, $\varepsilon > n^{-0.99}$) for both non-adaptive and adaptive algorithms [46, 63, 66], even with two-sided error. A simple reduction suggested in [109] shows that the same lower bound (up to a multiplicative factor depending on k) holds for any fixed $k \geq 2$. Thus, an appealing consequence of Theorem 4.1 is that the natural generalization of monotonicity testing, which considers forbidden monotone patterns of fixed length longer than 2, does not affect the dependence on n in the query complexity by more than a constant factor. Interestingly, Fischer [66] shows that for any adaptive algorithm for monotonicity testing on $f\colon [n] \to \mathbb{R}$ there is a non-adaptive algorithm which is at least as good in terms of query complexity (even if we only restrict ourselves to one-sided error algorithms). That is, adaptivity does not help at all for $k = 2$. In contrast, the separation between our $O(\log n)$ adaptive upper bound and the $\Omega\big((\log n)^{\lfloor \log_2 k \rfloor}\big)$ non-adaptive lower bound of [22] implies that this is no longer true for $k \geq 4$.

Harnessing adaptivity to improve algorithmic performance is a notoriously difficult problem in many branches of property testing, typically requiring a good structural understanding of the task at hand. In the context of testing for forbidden order patterns, non-adaptive algorithms are quite weak: see the next chapter for an extensive discussion. Prior to these results, the only case for which adaptive algorithms were known to outperform their non-adaptive counterparts was for patterns of length 3 in [109]. It is generally believed that there should be some separation between adaptive and non-adaptive testing algorithms for pattern detection; in fact, a conjecture of [109] suggests that for non-monotone patterns, the adaptive query complexity for testing π-freeness is polylogarithmic in n for *any* fixed-length π, an exponential improvement over non-adaptive algorithms. While this conjecture is still wide open, the result here is the first to show any kind of separation between adaptive and non-adaptive algorithms for patterns of length more than 3.

As an immediate consequence, Theorem 4.1 gives an optimal testing algorithm for the longest increasing subsequence (LIS) problem in a certain regime. The classical LIS problem asks to determine, given a sequence $f\colon [n] \to \mathbb{R}$, the maximum k for which f contains a length-k increasing subsequence. It is very closely related to other fundamental algorithmic problems in sequences, such as computing the edit distance, Ulam distance, or distance from monotonicity (for example, the latter equals n minus the LIS length), and was thoroughly investigated from the perspective of sublinear-time algorithms [2, 113, 123, 126] and streaming algorithms [62, 76, 87, 107, 125, 130]. In the property testing regime, the corresponding decision task is to distinguish between the case where f has LIS length at most k (where k is given as part of the input) and the case that f is ε-far from having such a LIS length. Theorem 4.1 in combination with the aforementioned lower bounds (which readily carry over to this setting) yield a tight bound on the query complexity of testing whether the LIS length is a constant.

Corollary 4.2. *Fix $2 \leq k \in \mathbb{N}$ and $\varepsilon > 0$. The query complexity of testing whether $f\colon [n] \to \mathbb{R}$ has LIS length at most k is $\Theta(\log n)$.*

4.1.1 Techniques

We now describe the intuition behind the proof of Theorem 4.1. There are two main technical components: 1) a strengthening of the structural result from the previous chapter, regarding splittable intervals and growing suffixes; and 2) new (adaptive) algorithmic components which lead to the $O(\log n)$-query algorithm.

Robustifying the structural decomposition. As mentioned above, there is an $O_{k,\varepsilon}(\log n)$-query non-adaptive algorithm for the growing suffixes case. Thus, in order to obtain an *adaptive* algorithm with such query complexity, it suffices to develop such an algorithm under the splittable intervals assumption. The splittable intervals condition, however, does not seem strong enough for our purposes. Recall that in the splittable intervals case, the algorithm in the previous chapter "guessed" the width w and lost a factor of $O(\log n)$ in the query complexity; the resulting analysis bounded the number of times the algorithm needed to guess by $\lfloor \log_2 k \rfloor$. In order to avoid the $O(\log n)$-factor loss more than once, one seemingly has to "identify", in some way, the correct width, and it is not clear how to do so effectively from the current structural theorem. In order to bypass this issue, we substantially strengthen the structural theorem. The stronger statement asserts that any $f \colon [n] \to \mathbb{R}$ that is ε-far from $(12\dots k)$-free satisfies either the growing suffixes condition, defined previously, or a *robust* version of the splittable intervals condition, defined shortly. Even though the algorithm will not be able to identify the correct width, the robust splittable intervals condition enables exploiting guesses for the width that are too large.

- **Robust splittable intervals:** there exist $c \in [k-1]$ and a collection of pairwise-disjoint intervals $I_1, \dots, I_s \subset [n]$ satisfying the same properties as in the "splittable intervals" setting described above (with slightly different dependence on ε and k in the $\Theta_{k,\varepsilon}(\cdot)$ term). Additionally, *any* interval $J \subset [n]$ which contains an interval I_j is itself far from $(12\dots k)$-free, i.e. it contains a collection of $\Omega_{k,\varepsilon}(|J|)$ disjoint $(12\dots k)$-copies.

The advantage of this robust condition is that fully containing a splittable interval suffices to conclude that the subsequence is $\Omega_{k,\varepsilon}(1)$-far from $(12\dots k)$-free (see Figure 4.1). As hinted above, a guess for the width which is too large results in an interval fully containing a splittable interval, and a lower bound on the fraction of length-k monotone subsequences inside the interval considered follows. We refer to this case as the "overshooting" case. We note that our proof of the "robust structural theorem" combines the basic structural theorem from the previous chapter, used as a black box, with additional combinatorial ideas.

Towards an algorithm. At a high level, the non-adaptive algorithms for this problem proceed in a recursive manner where each step tries to find the relevant width considered (which is one of $\Omega(\log n)$ options). Since their algorithms are non-adaptive, they consider all $\Omega(\log n)$ options in recursive steps, and hence, suffer a logarithmic factor with each step. Since our algorithm is adaptive, we want to choose *one* of the widths to recurse on. The algorithm ensures that the width considered is large enough. When the width chosen is not too much larger, our recursive step proceeds similarly to the non-adaptive algorithms; we call this the *fitting case*. However, the width considered may be too large; we call this case *overshooting*. In order to deal with overshooting, we utilize the robust structural theorem in a somewhat surprising manner to detect a $(12\dots k)$-copy.

We now expand on the above idea. As mentioned above, we may assume that f satisfies the

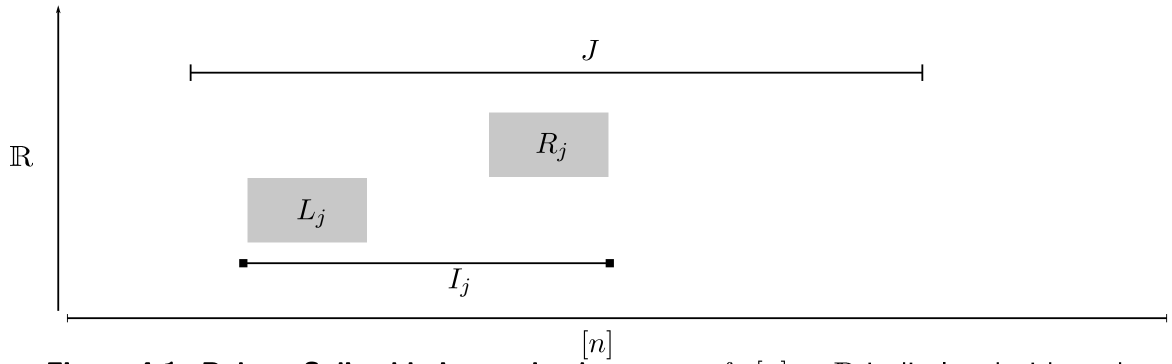


Figure 4.1: Robust Splittable Intervals. A sequence $f\colon [n] \to \mathbb{R}$ is displayed with a robust splittable interval I_j. Specifically, we have $c \in [k-1]$, L_j and R_j are two intervals such that L_j contains $\Omega_{k,\varepsilon}(|I_j|)$ many length-c monotone patterns and R_j contains $\Omega_{k,\varepsilon}(|I_j|)$ length-$(k-c)$ monotone subsequences. Furthermore, a subsequence in L_j may be combined with one in R_j to obtain a length-k monotone subsequence in I_j. The fact that I_j is a robust splittable interval is exemplified by the fact that any interval J such that $I_j \subset J$ contains $\Omega_{k,\varepsilon}(|J|)$ disjoint monotone subsequences of length k. We note that this holds even if $|J| \gg |I_j|$, so that most of the length-k monotone subsequences in J are not in I_j.

robust splittable condition. Sample, for $O_{k,\varepsilon}(1)$ repetitions, an index $\boldsymbol{x} \in [n]$ uniformly at random, and for each $t \in [\log n]$, a random index $\boldsymbol{y}_t \in [\boldsymbol{x} + 2^{t-1}, \boldsymbol{x} + 2^t]$. Consider the following event:

> The index $\boldsymbol{x}$ is a (sufficiently well-behaved)[1] first element in some $(12\ldots k)$-pattern falling in some robust splittable interval I_j, and for $t^* \in [\log n]$ satisfying $|I_j| \leq 2^{t^*} \leq 2|I_j|$, $\boldsymbol{y}_{t^*}$ is a (well-behaved) $(c+1)$-th element in some $(12\ldots k)$-pattern falling in I_j.

We claim that, with high probability, the above event occurs for at least one choice of $\boldsymbol{x}$, and that when this event does occur, the algorithm can be applied recursively without incurring a multiplicative logarithmic factor. Indeed, suppose that the above holds for some $\boldsymbol{x}$.[2] We set $\boldsymbol{y}$ to be $\boldsymbol{y}_t$, where t is the largest such that $f(\boldsymbol{x}) < f(\boldsymbol{y}_t)$ holds, and notice in particular that $t \geq t^*$. This means that $\boldsymbol{x} < \boldsymbol{y}$ and $f(\boldsymbol{x}) < f(\boldsymbol{y})$.

The fitting component The *fitting case* occurs when t (achieving the maximum above) is roughly the same as t^*. To handle this case, we recurse by finding a $(12\ldots c)$-patterns in L_j, and $(12\ldots(k-c))$-pattern in R_j. At a high level, if one takes $\Theta_{k,\varepsilon}(1)$ independent uniform samples $\boldsymbol{z}$

[1]Recall that, in the non-adaptive algorithm, we hoped to hit a "1-entry" x whose value $f(x)$ is no higher than some suitable median value; the "well-behaved" requirements are of similar flavor, and do not incur more than a constant overhead on the query complexity.

[2]More precisely, our algorithm runs this procedure for any of our choices of $\boldsymbol{x}$, without "knowing" which of them satisfies the above event. Since the total number of choices is $O_{k,\varepsilon}(1)$, this incurs only a constant overhead.

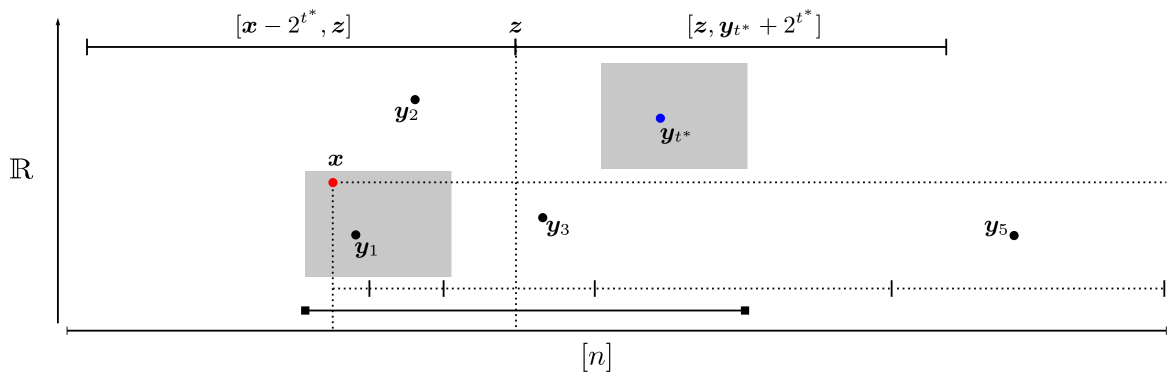


Figure 4.2: The fitting case. A sequence $f\colon [n] \to \mathbb{R}$ which falls in the splittable intervals case, and the event that we refer to as the "fitting case" is presented. Specifically, the algorithm proceeds by first sampling $\boldsymbol{x} \sim [n]$ which falls in a robust splittable interval, and corresponds to the first index of some length-k monotone subsequence. The algorithm then considers sampling one $\boldsymbol{y}_t$ uniformly from the interval for $[\boldsymbol{x} + 2^{t-1}, \boldsymbol{x} + 2^t]$. The algorithm looks for the largest $t \leq \log_2 n$ for which $f(\boldsymbol{x}) < f(\boldsymbol{y}_t)$. The relevant event for the fitting case is that for the appropriate width containing R_j, corresponding to t^*, $\boldsymbol{y}_{t^*} \in R_j$ and corresponds to the $(c+1)$-th index in a length-k monotone subsequence contained in I_j. In the fitting case, t is not too much larger than t^*; in particular, the figure shows a case when $t = t^*$. The algorithm then samples $\boldsymbol{z} \sim [\boldsymbol{x}, \boldsymbol{y}_t]$ which we show falls in M_j with constant probability, and the the algorithm recurses to search for a $(12\dots c)$-pattern on the interval $[\boldsymbol{x} - 2^{t^*}.\boldsymbol{z}]$ and a $(12\dots(k-c))$-pattern on the interval $[\boldsymbol{z}, \boldsymbol{y}_t + 2^t]$.

from $[\boldsymbol{x}, \boldsymbol{y}]$, then one of them is likely to fall in the middle part M_j of I_j, so that $L_j \subset [\boldsymbol{x} - 2^t, \boldsymbol{z}]$ and $R_j \subset [\boldsymbol{z}, \boldsymbol{y} + 2^t]$, allowing us to proceed recursively. This is conceptually similar to the algorithms of [109] and [22], except that the recursion occurs only on one width, namely t, and it therefore does not lose multiplicative logarithmic factors as in the previous approaches.

The overshooting component. The other case, of *overshooting*, occurs when t is significantly larger than t^*. We expand on the main ideas here in more detail. The strong guarantee given by the robust splittable intervals condition adds a "for all" element into the structural characterization, which is able to treat the problem posed by overshooting. When t is significantly larger than $\log |I_j|$, there exist $k-2$ intervals $J_1, \dots, J_{k-2} \subset [\boldsymbol{x}, \boldsymbol{y}]$ satisfying the following conditions:

- J_1 lies immediately after the interval I_j (recall that I_j is the interval containing $\boldsymbol{x}$).
- J_{i+1} lies immediately after J_i, for any $i \in [k-3]$.
- $|J_1| = |I_j| \cdot \alpha_{k,\varepsilon}$ and $|J_{i+1}| = |J_i| \cdot \alpha_{k,\varepsilon}$ for any $i \in [k-3]$, for some large enough $\alpha_{k,\varepsilon} > 1$.

The surprising consequence of the robust splittable intervals condition is that even though the intervals $J_1, \dots, J_{k-2}$ are disjoint from I_j, for every $i \in [k-2]$, the interval J_i contains $\Omega_{k,\varepsilon}(|J_i|)$ disjoint length-k monotone subsequences. At a high level, the argument proceeds by considering, for any $i \in [k-2]$, set J_i' to be the shortest interval containing both I_j and J_i. The robust splittable intervals condition asserts that (since each J_i' contains the splittable interval I_j) the number of disjoint $(12\dots k)$-copies in J_i' is proportional to $|J_i'|$. Provided that $\alpha_{k,\varepsilon}$ is large enough, this means that $J_i = J_i' \setminus J_{i-1}'$ also contains a collection $\mathcal{T}_i$ of $\Omega_{k,\varepsilon}(|J_i|)$ disjoint $(12\dots k)$-copies. We now define two sets $\mathcal{A}_i$ and $\mathcal{B}_i$ as follows. Let $\mathcal{A}_i$ be the collection of prefixes $(a_1, \dots, a_{i+1})$ of k-tuples from $\mathcal{T}_i$ with $f(a_{i+1}) < f(y)$, and let $\mathcal{B}_i$ be the collection of suffixes $(a_{i+1}, \dots, a_k)$ of k-tuples from $\mathcal{T}_i$ with $f(a_{i+1}) \geq f(y)$. As $|\mathcal{T}_i| = |\mathcal{A}_i| + |\mathcal{B}_i|$, one of $\mathcal{A}_i$ and $\mathcal{B}_i$ is large (i.e. has size at least $\Omega_{k,\varepsilon}(|J_i|)$).

This seemingly innocent combinatorial idea can be exploited non-trivially to find a $(12\dots k)$-copy. Specifically, the algorithm to handle overshooting aims to (recursively) find shorter increasing subsequences in $J_1, \dots, J_{k-2}$, with the hope of combining them together into a $(12\dots k)$-copy. Concretely, for any $i \in [k-2]$, we make two recursive calls of our algorithm on J_i: one for a $(k-i)$-increasing subsequence in J_i whose values are at least $f(y)$,[3] and a second for an $(i+1)$-increasing subsequence in J_i, with values smaller than $f(y)$. By induction, the first recursive call succeeds with good probability if $|\mathcal{A}_i|$ is large, while the second call succeeds with good probability if $|\mathcal{B}_i|$ is large. Since for any i either $|\mathcal{A}_i|$ or $|\mathcal{B}_i|$ must be large, at least one of the following must hold.

- $\mathcal{B}_1$ is large. In this case we are likely to find a length-$(k-1)$ monotone pattern in J_1 with values at least $f(y) > f(x)$, which combines with $\boldsymbol{x}$ to form a length-k monotone pattern.

- $\mathcal{A}_{k-2}$ is large. Here we are likely to find a length-$(k-1)$ monotone pattern in J_{k-2} whose values lie below $f(y)$, which combines with $\boldsymbol{y}$ to form a length-k monotone pattern.

- There exists $i \in [k-3]$ where both $\mathcal{A}_i$ and $\mathcal{B}_{i+1}$ are large. Here we will find, with good probability, a length-$(i+1)$ monotone pattern in J_i with values below $f(y)$, and a length-$(k-i-1)$ monotone pattern in J_{i+1} with values above $f(y)$; together these two patterns combine to form a $(12\dots k)$-pattern.

In all cases, a k-increasing subsequence is found with good probability. See Figure 4.3 for an example.

Query complexity. Finally, for the query complexity, our algorithm (which runs both the "fitting" component and the "overshooting" component, to address both cases) makes $O_{k,\varepsilon}(\log n)$ queries in total. This holds as each call makes $O_{k,\varepsilon}(\log n)$ queries in itself and $O_{k,\varepsilon}(1)$ additional calls recursively, where the recursion depth is bounded by k.

[3]Technically speaking, our algorithm can be configured to only look for increasing subsequences whose values lie in some range; we use this to make sure that shorter increasing subsequences obtained from the recursive calls of the algorithm can eventually be concatenated into a valid length-k one.

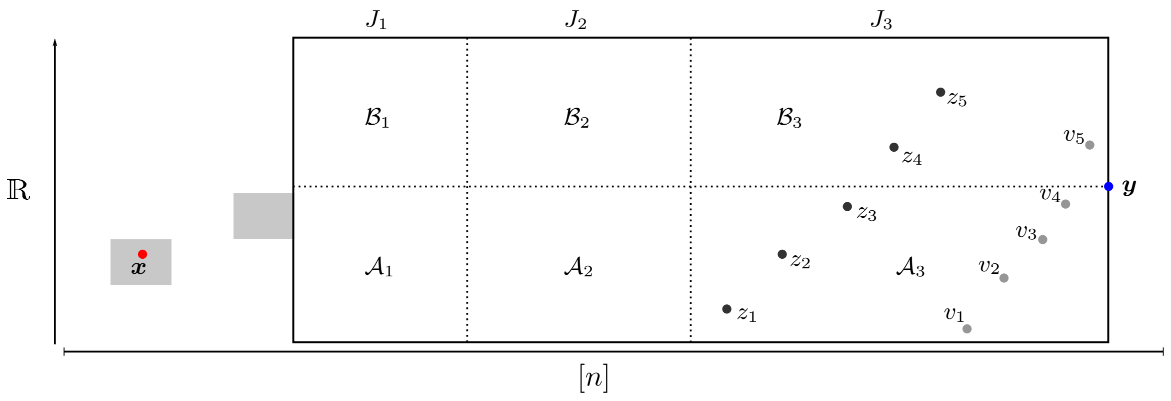


Figure 4.3: The overshooting case. A sequence $f\colon [n] \to \mathbb{R}$ which falls in the splittable intervals case for $k = 5$. We consider the "overshooting case". Specifically, we consider a case when the algorithm samples $\boldsymbol{x} \sim [n]$ which falls in a splittable interval, but when we sample $\boldsymbol{y}_t \sim [\boldsymbol{x} + 2^{t-1}, \boldsymbol{x} + 2^t]$ for all $t \in [\log_2 n]$, the largest $t \in [\log_2 n]$ where $f(\boldsymbol{y}_t) > f(\boldsymbol{x})$ falls very far from the splittable interval. In the figure, we omit the points $\boldsymbol{y}_1, \dots, \boldsymbol{y}_t$ and let $\boldsymbol{y}$ be the largest point where $f(\boldsymbol{x}) < f(\boldsymbol{y})$. The "overshooting case" considers the point where at least $k - 2$ geometrically increasing intervals fit between $\boldsymbol{x}$ and $\boldsymbol{y}$; these are displayed as J_1, J_2, and J_3; by virtue of the splittable interval being robust, each J_i contains $\Omega_{k,\varepsilon}(|J_i|)$ disjoint length-k monotone subsequences. $\mathcal{A}_i$ contains those length-k monotone subsequences where the $(i+1)$-th index is above $f(\boldsymbol{y})$ and $\mathcal{B}_i$ contains those whose $(i+1)$-th index is below $f(\boldsymbol{y})$. As an example, $(z_1, z_2, z_3, z_4, z_4) \in \mathcal{B}_4$ and $(v_1, v_2, v_3, v_4, v_5) \in \mathcal{A}_4$. The crucial properties are: (i) for all $i \in [k-2]$ any $(12 \dots i)$-pattern in $\mathcal{A}_i$ and any $(12 \dots (k-i))$-pattern in $\mathcal{B}_{i+1}$ may be combined into a $(12 \dots k)$-pattern, (ii) any $(12 \dots (k-1))$-pattern in $\mathcal{B}_1$ may be combined with $\boldsymbol{x}$ since $f(\boldsymbol{y}) > f(\boldsymbol{x})$, and (iii) any $(12 \dots (k-1))$-pattern in $\mathcal{A}_4$ may be combined with $\boldsymbol{y}$. The reasoning may proceed as follows: if $|\mathcal{B}_1|$ is large, we find a $(12 \dots (k-1))$-pattern and combine it with $\boldsymbol{x}$; so, assume $|\mathcal{B}_1|$ is small, which implies $|\mathcal{A}_1|$ must be large. If $|\mathcal{B}_2|$ is large, then a (12)-pattern from $\mathcal{A}_1$ and a $(12 \dots (k-2))$-pattern from $\mathcal{B}_2$ may be combined; so assume $|\mathcal{B}_2|$ is small which implies $|\mathcal{A}_2|$ is large, ... Eventually, we deduce that we may assume $|\mathcal{A}_{k-2}|$ is large, and a $(12 \dots (k-1))$-pattern in $\mathcal{A}_{k-2}$ may be combined with $\boldsymbol{y}$.

Notation The notation used here generally extends that of Chapter 3. In many cases, we think of augmenting the image of our input function $f : [n] \to \mathbb{R}$ restricted to an interval $I \subseteq [n]$ with a special character $*$ to consider $f\colon I \to \mathbb{R} \cup \{*\}$. The character $*$ can be thought of as a *masking* operation: In many cases, we will only be interested in entries x of f so that $f(x)$ lies in some prescribed (known in advance) range of values $R \subseteq \mathbb{R}$, so that entries outside this range will be marked by $*$. Whenever the algorithm queries $f(x)$ and observes $*$, it will interpret this as an incomparable value (with respect to ordering) in $\mathbb{R}$. As a result, $*$-values will never be part of monotone subsequences. We note that augmenting the image with $*$ is unnecessary when considering non-adaptive algorithms. We say that for a fixed $f\colon I \to \mathbb{R} \cup \{*\}$, the set T is a collection of disjoint monotone subsequences of length k if it consists of tuples $(i_1, \ldots, i_k) \in I^k$, where $i_1 < \cdots < i_k$ and $f(i_1) < \cdots < f(i_k)$ (in particular, $f(i_1), \ldots, f(i_k) \neq *$), and furthermore, for any two tuples $(i_1, \ldots, i_k)$ and $(i'_1, \ldots, i'_k)$, their intersection (as sets) is empty. We also denote $E(T)$ as the union of indices in k-tuples of T, i.e., $E(T) = \cup_{(i_1,\ldots,i_k)\in T}\{i_1, \ldots, i_k\}$.

4.2 Stronger Structural Dichotomy

In this section, we establish the structural foundations – specifically, the *growing suffixes* versus *robust splittable intervals* dichotomy – lying at the heart of our adaptive algorithm. Recall the definition of the growing suffix and splittable intervals settings, as given in the previous chapter (see Definitions 3.6 and 3.7).

The following theorem is a restatement of the growing suffixes versus (non-robust) splittable intervals dichotomy from the previous chapter. There, the theorem is stated with respect to two parameters, k, k_0; for our purpose it suffices to set $k_0 = k$. Also, here we allow f to take the value $*$, which is not the case in the previous chapter. Nevertheless, as the proof there takes into account only the elements of a given family T^0 of disjoint length-k increasing subsequences, which in particular are non-$*$ elements, the same proof would work here.

Theorem 4.3 (Simplified form of Theorem 3.8)**.** *Let $k, n \in \mathbb{N}$, $\varepsilon \in (0, 1)$, and $C > 0$, and let $I \subseteq [n]$ be an interval. Let $f\colon I \to \mathbb{R} \cup \{*\}$ be a function and let $T^0 \subseteq I^k$ be a set of at least $\varepsilon|I|$ disjoint monotone subsequences of f of length k. Then there exist $\alpha \in (0, 1)$ and $p > 0$ satisfying $\alpha \geq \Omega(\varepsilon/k^5)$ and $p \leq \mathrm{poly}(k \log(1/\varepsilon))$ such that at least one of the following conditions holds.*

1. ***Growing suffixes:*** *There exists a set $H \subseteq [n]$, of indices that start an $(\alpha, Ck\alpha)$-growing suffix, satisfying $\alpha|H| \geq (\varepsilon/p)n$.*

2. ***Splittable intervals (non-robust):*** *There exist an integer c with $1 \leq c < k$, a set T, with $E(T) \subseteq E(T^0)$, of disjoint length-k monotone subsequences, and a $(c, 1/(6k), \alpha)$-splittable collection of T, consisting of disjoint interval-tuple pairs $(I_1, T_1), \ldots, (I_s, T_s)$, such that*

$$\alpha \sum_{h=1}^{s} |I_h| \geq |T^0|/p. \tag{4.1}$$

As argued in Section 4.1.1, the splittable intervals condition does not seem strong enough by itself to be useful for adaptive algorithms. Therefore, we next aim to establish a stronger structural dichotomy, asserting that f satisfies either the growing suffixes condition, or a *robust* version of the splittable intervals condition. The next lemma will imply that the growing suffixes condition can be robustified by merely throwing away a subset of "bad" splittable intervals.

Lemma 4.4. *Let $\alpha \in (0,1)$ and let $I \subset \mathbb{N}$ be an interval. Suppose that $I_1, \dots, I_s \subset I$ are disjoint intervals such that $\sum_{h=1}^{s} |I_h| \geq \alpha |I|$. Then there exists a set $G \subset [s]$ such that*

$$\sum_{h \in G} |I_h| \geq (\alpha/4)|I|,$$

and for every interval $J \subset I$ that contains an interval I_h with $h \in G$,

$$\sum_{h \in [s]\colon I_h \subset J} |I_h| \geq (\alpha/4)|J|.$$

Proof. Let $B \subseteq [s]$ be the set of indices h for which there is an interval $J_h \supseteq I_h$ satisfying $\sum_{h \in [s] : I_h \subseteq J} |I_h| < (\alpha/4)|J|$. For each $h \in B$ fix such a containing interval $J(I_h)$.

Let $\mathcal{J}$ be a minimal subset of $\{J(I_h) : h \in B\}$ with the following property: for any $h \in B$ there exists $J \in \mathcal{J}$ containing I_h. Such a minimal subset clearly exists, since $\{J(I_h) : h \in B\}$ itself satisfies this property (but is not necessarily minimal). The next claim asserts that no vertex is covered more than three times by sets in $\mathcal{J}$.

Claim 4.5. *Every element $x \in I$ is contained in at most three intervals from $\mathcal{J}$.*

Proof. The proof follows from the minimality of $\mathcal{J}$. Consider first the case where $x \in I_{h^*}$ for some $h^* \in B$. Let $J_L = [a_L, b_L]$ be an interval from $\mathcal{J}$ that contains x, and whose left-most element a_L is furthest to the left among all intervals from $\mathcal{J}$ that contain x; pick $J_R = [a_R, b_R]$ symmetrically, with b_R being furthest possible to the right; and let $J_M = [a_M, b_M]$ be an interval from $\mathcal{J}$ that contains I_h. We claim that $\mathcal{J}$ does not have any other intervals that contain x. Suppose, to the contrary, that there exists $J = [a, b] \in \mathcal{J}$ containing x where $J \neq J_L, J_R, J_M$; note that by definition of J_L and J_M, $a_L \leq a$ and $b_R \geq b$.

We claim that $\mathcal{J} \setminus \{J\}$ covers all intervals I_h with $h \in [B]$; it suffices to show that for any $h \in B$ such that $I_h \subset J$, one of the intervals J_L, J_R, J_M covers I_h. Consider $h \in B$ such that $I_h \subset J$, and write $I_h = [c, d]$. If $h = h^*$, then $I_h \subset J_M$. If I_h lies to the left of I_{h^*}, then $d < x \leq b_L$, and $c \geq a \geq a_L$, so $I_h \subseteq J_L$. Similarly, if I_h lies to the right of I_h, then $I_h \subseteq J_R$. It follows that, indeed, intervals from $\mathcal{J} \setminus \{J\}$ cover all intervals in $\{I_h : h \in B\}$, contradicting the minimality of $\mathcal{J}$.

Now, if x is not contained in any interval of I_h with $h \in B$, then we can show similarly that there are at most two intervals from $\mathcal{J}$ that contain x, by defining J_L and J_R as above. □

Let U be the union of intervals from $\mathcal{J}$. In light of the above claim,

$$\sum_{h\in B}|I_h| \le \sum_{J\in\mathcal{J}}\left(\sum_{h\in[s]\colon I_h\subseteq J}|I_h|\right) < \frac{\alpha}{4}\cdot\sum_{J\in\mathcal{J}}|J| \le \frac{3\alpha}{4}\cdot|U| \le \frac{3\alpha}{4}\cdot|I|,$$

where the first inequality holds because each I_h with $h \in B$ is covered by an interval in $\mathcal{J}$; the second inequality follows from the definition of B, as $\mathcal{J}$ consists of sets $J(I_h)$ with $h \in B$; the third inequality follows from the claim; and the last one holds because $U \subset I$. Finally, let $G = [s] \setminus [B]$. By assumption on $\sum_h |I_h|$ and the previous line,

$$\sum_{h\in G}|I_h| = \sum_{h\in[s]}|I_h| - \sum_{h\in B}|I_h| \ge \alpha|I| - \frac{3\alpha}{4}\cdot|I| = \frac{\alpha}{4}\cdot|I|,$$

and every interval J that contains an interval I_h with $h \in G$ satisfies $\sum_{h\in[s]\colon I_h\subset J}|I_h| \ge (\alpha/4)|J|$, as required. □

The robust version of the structural dichotomy is stated below; the proof follows easily from the basic structural dichotomy in combination with the last lemma.

Theorem 4.6 (Robust structural theorem)**.** *Let $k, n \in \mathbb{N}$, $\varepsilon \in (0,1)$, and $C > 0$, and let $I \subseteq [n]$ be an interval. Let $f\colon I \to \mathbb{R}\cup\{*\}$ be an array and let $T^0 \subseteq I^k$ be a set of at least $\varepsilon|I|$ disjoint length-k monotone subsequences of f. Then there exist $\alpha \in (0,1)$ and $p > 0$ with $\alpha \ge \Omega(\varepsilon/k^5)$ and $p \le \operatorname{poly}(k\log(1/\varepsilon))$ such that at least one of the following holds.*

1. ***Growing suffixes:*** *There exists a set $H \subseteq [n]$, of indices that start an $(\alpha, Ck\alpha)$-growing suffix, satisfying $\alpha|H| \ge (\varepsilon/p)n$.*

2. ***Robust splittable intervals:*** *There exist an integer c with $1 \le c < k$, a set T, with $E(T) \subseteq E(T^0)$, of disjoint length-k monotone subsequences, and a $(c, 1/(6k), \alpha)$-splittable collection of T, consisting of disjoint interval-tuple pairs $(I_1, T_1), \ldots, (I_s, T_s)$, such that*

$$\alpha\sum_{h=1}^{s}|I_h| \ge (\varepsilon/p)|I|, \tag{4.2}$$

Moreover, if $J \subset I$ is an interval where $J \supset I_h$ for some $h \in [s]$, then J contains at least $(\varepsilon/p)|J|$ disjoint $(12\ldots k)$-patterns from T^0.

Proof. Apply Theorem 4.3. Let $\alpha^* \in (0,1)$ and p^* be parameters such that $\alpha^* \ge \Omega(\varepsilon/k^5)$ and $p^* \le \operatorname{poly}(k\log(1/\varepsilon))$, as guaranteed by the theorem. Set $\alpha = \alpha^*$ and $p = 4p^*$. If Condition 1 holds in the application of Theorem 4.3, then the analogous growing suffix condition in Theorem 4.6 clearly holds. So suppose that Condition 2 in Theorem 4.3 holds, and let c and $(I_1, T_1), \ldots, (I_s, T_s)$ be as guaranteed there. In particular, we have $\sum_{h=1}^{s}|I_h| \ge (1/p^*\alpha^*)|T^0|$. By Lemma 4.4, there is a subset $G \subset [s]$ such that $\sum_{h\in G}|I_h| \ge (1/4p^*\alpha^*)|T^0| \ge (\varepsilon/4p^*\alpha^*)|I| = (\varepsilon/p\alpha)|I|$; and, for every interval J in I that contains an interval I_h with $h \in [G]$, $\sum_{h\in[s]:\, I_h\subset J} |I_h| \geq (\varepsilon/4p^*\alpha^*)|J|$. Since each I_h contains at least $\alpha^*|I_h|$ disjoint length-k increasing subsequences, it follows that J contains at least $(\varepsilon/4p^*)|J| = (\varepsilon/p)|J|$ length-k increasing subsequences. Taking T to be the union of T_h over $h \in G$, along with the pairs (I_h, T_h) with $h \in G$, we obtain the required robust splittable intervals. □

4.3 The Algorithm

Our aim in this section is to prove the existence of a randomized algorithm, $\texttt{Find-Monotone}_k(f, \varepsilon, \delta)$, that receives as input a function $f\colon I \to \mathbb{R}\cup\{*\}$ (where $I \subset \mathbb{N}$ is an interval), and parameters $\varepsilon, \delta \in (0, 1)$, and satisfies the following: if f contains $\varepsilon|I|$ disjoint $(12\ldots k)$-patterns, then the algorithm outputs such a pattern with probability at least $1-\delta$; and the running time of the algorithm is $O_{k,\varepsilon}(\log n)$. To this end, we describe such an algorithm in Figure 4.6 below. This algorithm uses three subroutines: `Sample-Suffix`, `Find-Within-Interval`, and `Find-Good-Split`, the first of which is given in the previous chapter, and the latter two are described below, in Figures 4.4 and 4.5. The majority of the section is devoted to the proof that `Find-Monotone` indeed outputs a $(12\ldots k)$-pattern with high probability as claimed. Specifically, we shall prove the following theorem.

Theorem 4.7. *Let $k \in \mathbb{N}$. The randomized algorithm $\texttt{Find-Monotone}_k(f, \varepsilon, \delta)$, described in Figure 4.6, satisfies the following. Given a function $f\colon I \to \mathbb{R}\cup\{*\}$ and parameters $\varepsilon, \delta \in (0, 1)$, if f contains at least $\varepsilon|I|$ disjoint $(12\ldots k)$-patterns, then $\texttt{Find-Monotone}_k(f, \varepsilon, \delta)$ outputs a $(12\ldots k)$-pattern of f with probability at least $1-\delta$.*

Our proof proceeds by induction on k. It relies on a slightly modified version of Lemmas 3.20, along with two new claims, Lemmas 4.10, 4.11. The proofs of the latter two assume that Theorem 4.7 holds for smaller k. We first state and prove these lemmas, and then we prove Theorem 4.7.

To complete the picture, we need to upper-bound the query complexity and running time of `Find-Monotone`. We do this in the following lemma, whose proof we delay to the end of the section.

Lemma 4.8. *Let $f\colon I \to \mathbb{R}\cup\{*\}$, where I is an interval of length at most n. The query complexity and running time of $\texttt{Find-Monotone}_k(f, \varepsilon, \delta)$ are at most*

$$\left(k^k \cdot (\log(1/\varepsilon))^k \cdot \frac{1}{\varepsilon} \cdot \log(1/\delta)\right)^{O(k)} \cdot \log n.$$

The `Sample-Suffix` Sub-Routine

We restate Lemma 3.20 which gives the $\texttt{Sample-Suffix}_k$ subroutine, with a few adaptations to fit our needs.

Lemma 4.9 (Restatement of Lemma 3.20)**.** *Consider any fixed value $k \in \mathbb{N}$, and let $C > 0$ be a large enough constant. There exists a non-adaptive and randomized algorithm* $\texttt{Sample-Suffix}_k(f, \varepsilon, \delta)$ *which takes three inputs: query access to a function $f \colon I \to \mathbb{R} \cup \{*\}$, where $I \subset \mathbb{N}$ is an interval, a parameter $\varepsilon \in (0, 1)$, and an error probability bound $\delta \in (0, 1)$. Suppose there exists $\alpha \in (0, 1)$, and a set $H \subseteq I$ of $(\alpha, Ck\alpha)$-growing suffixes of f satisfying $\alpha|H| \geq \varepsilon|I|$. Then,* $\texttt{Sample-Suffix}_k(f, \varepsilon, \delta)$ *finds a length-k monotone subsequence of f with probability at least $1 - \delta$. The query complexity of* $\texttt{Sample-Suffix}_k(f, \varepsilon, \delta)$ *is at most*

$$\log(1/\delta) \cdot \text{polylog}(1/\varepsilon) \cdot \frac{1}{\varepsilon} \cdot \log n.$$

There are two differences between this statement and Lemma 3.20. First, here we have error probability δ, whereas in Lemma 3.20 the error probability is $1/10$. In order to achieve error probability δ, we perform $O(\log(1/\delta))$ independent repetitions of $\texttt{Sample-Suffix}_k$, as described in the previous chapter. These are reflected in the query complexity. The second difference is that we consider functions $f \colon I \to \mathbb{R} \cup \{*\}$. Inspecting the proof of Lemma 3.20, one can see that $\texttt{Sample-Suffix}_k$ is guaranteed to output, with high probability, $(12 \ldots k)$-patterns whose indices are specified in Definition 3.6. Since the algorithm is non-adaptive, enforcing that indices not partaking in growing suffices not be used (by making them $*$) does not affect that analysis.

Handling Overshooting: The `Find-Within-Interval` Sub-Routine

In this section, we describe the `Find-Within-Interval` subroutine, addressing the overshooting case as explained in Section 4.1.1.

As the algorithm may appear un-intuitive, let us remind the reader of the setup in which this subroutine is relevant (see also Section 4.1.1). By Theorem 4.6, either the growing suffixes condition or the splittable intervals condition hold. The former case is handled by Lemma 4.9, so we assume that the latter holds. Now assume that we sampled an element $\boldsymbol{x}$ which is the first element of a length-c increasing subsequence from a set L_i as described in Definition 3.7. We then sample, uniformly at random, elements $\boldsymbol{y}$ from $[\boldsymbol{x}, \boldsymbol{x} + 2^t]$. The splittable intervals condition implies that we will find, with high probability, an element $\boldsymbol{y}$ which is the last element of a length-$(k-c)$ increasing subsequence from R_i. In particular, $f(\boldsymbol{y}) > f(\boldsymbol{x})$. However, even if we did indeed sample such $\boldsymbol{y}$, we may have sampled many other values of $\boldsymbol{y}'$ with $f(\boldsymbol{y}') > f(\boldsymbol{x})$, and we do not know of a way of determining which of these values is the "correct" one. Instead, we take $\boldsymbol{y}_0$ to be the largest sampled $\boldsymbol{y}'$ such that $f(\boldsymbol{y}') > f(\boldsymbol{x})$. The case where $\boldsymbol{y}_0$ is close to $\boldsymbol{y}$ is taken care of by Lemma 4.10, so we assume that $\boldsymbol{y}_0$ is much larger than $\boldsymbol{y}$.

We now have elements $\boldsymbol{x}$ and $\boldsymbol{y}_0$, and all that we know is that they contain a large portion of an interval I_i from the splittable intervals condition. It is not hard to see (this is shown in the proof of Theorem 4.7) that $[\boldsymbol{x}, \boldsymbol{y}_0]$ can be partitioned into $k-2$ intervals $J_1, \ldots, J_{k-2}$, each of which contains many disjoint length-k increasing subsequences. To continue, out only hope is use the induction hypothesis to find shorter increasing subsequences in the intervals. For example, if there are many

Subroutine `Find-Within-Interval`$_k(f, \varepsilon, \delta, x, y, \mathcal{J})$.

Input: Query access to a function $f\colon I \to \mathbb{R} \cup \{*\}$, parameters $\varepsilon, \delta \in (0,1)$, two inputs $x, y \in I$ where $x < y$ and $f(x) < f(y)$, and $\mathcal{J} = (J_1, \ldots, J_{k-2})$ which is a collection of disjoint intervals appearing in order inside $[x, y]$.

Output: a sequence $i_1 < \ldots < i_k$ with $f(i_1) < \ldots < f(i_k)$, or fail.

1. For every $\kappa \in [k-2]$, let $f_\kappa, f'_\kappa\colon J_\kappa \to \mathbb{R} \cup \{*\}$ be given by:

$$f_\kappa(i) = \begin{cases} f(i) & f(i) < f(y) \\ * & \text{o.w.} \end{cases} \qquad \text{and} \qquad f'_\kappa(i) = \begin{cases} f(i) & f(i) \geq f(y) \\ * & \text{o.w.} \end{cases} . \tag{4.3}$$

2. Call `Find-Monotone`$_{\kappa+1}(f_\kappa, \varepsilon/2, \delta/(2k))$ for every $\kappa \in [k-2]$.

3. Call `Find-Monotone`$_{k-\kappa}(f'_\kappa, \varepsilon/2, \delta/(2k))$ for every $\kappa \in [k-2]$.

4. Consider the set of all indices that are output in Lines 2 and 3, together with x and y. If there is a length-k increasing subsequence among these indices, output it. Otherwise, output fail.

Figure 4.4: Description of the `Find-Within-Interval` subroutine.

disjoint length-$(k-1)$ increasing subsequences in J_1 that lie above $\boldsymbol{x}$, then one such subsequence is likely to be detected by a recursive call to the main algorithm, and together with $\boldsymbol{x}$ it will form a length-k increasing subsequence. If there are few such length-$(k-1)$ subsequences, this means that there are many disjoint length-2 increasing subsequences in J_1 that lie below $\boldsymbol{x}$ (because for every length-k increasing subsequence, either its $(k-1)$-suffix lies above $\boldsymbol{x}$, or its 2-prefix lies above $\boldsymbol{x}$). We can then use a recursive call to detect such a sequence, and hope to complete it to a length-k subsequence using a length-$(k-2)$ subsequence from J_2 that lies above $\boldsymbol{x}$. Continuing with this logic, it follows that with high probability we can find an increasing subsequence of length k using $\boldsymbol{x}$ and J_1, J_i and J_{i+1} for some i, or J_{k-2} and $\boldsymbol{y}_0$.

Lemma 4.10. *Consider the randomized algorithm,* `Find-Within-Interval`$_k(f, \varepsilon, \delta, x, y, \mathcal{J})$*, described in Figure 4.4, which takes six inputs:*

- *Query access to a function $f\colon I \to \mathbb{R} \cup \{*\}$,*
- *Two parameters $\varepsilon, \delta \in (0,1)$,*
- *Two points $x, y \in I$ where $x < y$ and $f(x) < f(y)$, and*
- *A collection $\mathcal{J} = (J_1, \ldots, J_{k-2})$ of $k-2$ disjoint intervals that appear in order (i.e., J_κ comes before $J_{\kappa+1}$) within the interval $[x, y]$,*

and outputs either a length-k increasing subsequence of f, or fail.

Suppose that for every $\kappa \in [k-2]$, the function $f|_{J_\kappa}\colon J_\kappa \to \mathbb{R} \cup \{*\}$, contains $\varepsilon|J_\kappa|$ disjoint $(12\ldots k)$-patterns. Then, assuming that Theorem 4.7 holds for every k' with $1 \le k' < k$, `Find-Within-Interval`$_k(f, \varepsilon, \delta, x, y, \mathcal{J})$ *outputs a length-k monotone subsequence of f with probability at least $1-\delta$.*

Proof. For each $\kappa \in [k-2]$, let $\mathcal{C}_\kappa$ be a collection of at least $\varepsilon|J_\kappa|$ disjoint $(12\ldots k)$-patterns in J_κ. We form the following two collections, of suffixes and prefixes of $(12\ldots k)$-patterns in $\mathcal{C}_\kappa$.

$$\mathcal{A}_\kappa = \{(i_1, \ldots, i_{\kappa+1}) : (i_1, \ldots, i_{\kappa+1}) \text{ is a prefix of a } k\text{-tuple from } \mathcal{C}_k, \text{ and } f(i_{\kappa+1}) < f(y)\}$$
$$\mathcal{B}_\kappa = \{(i_{\kappa+1}, \ldots, i_k) : (i_{\kappa+1}, \ldots, i_k) \text{ is a suffix of a } k\text{-tuple from } \mathcal{C}_k, \text{ and } f(i_{\kappa+1}) \ge f(y)\}$$

Note that for each $(12\ldots k)$-pattern in $\mathcal{C}_\kappa$, either its $(\kappa+1)$-prefix is in $\mathcal{A}_\kappa$, or its $(k-\kappa)$-suffix is in $\mathcal{B}_\kappa$. Thus, at least one of $\mathcal{A}_\kappa$ and $\mathcal{B}_\kappa$ has size at least $(\varepsilon/2)|J_\kappa|$. Say that J_κ is of type-1 if $|\mathcal{A}_\kappa| \ge (\varepsilon/2)|J_\kappa|$, and otherwise say that J_κ is of type-2 (in which case $|\mathcal{B}_\kappa| \ge (\varepsilon/2)|J_\kappa|$).

Now, if J_κ is of type-1, then Line 2, called with κ, will find a $(12\ldots(\kappa+1))$-pattern with probability at least $1-\delta/(2k)$, by Theorem 4.7 for $\kappa+1 < k$ (namely, the inductive hypothesis) and the lower bound on $|\mathcal{A}_\kappa|$. On the other hand, if J_κ is of type-2, Line 3 will output a $(12\ldots(k-\kappa))$-pattern with probability at least $1-\delta/(2k)$, due to the inductive hypothesis and the lower bound on $|\mathcal{B}_\kappa|$. Thus, by a union bound, with probability at least $1-\delta$, Line 2 outputs a pattern whenever J_κ is of type-1, and Line 3 outputs a pattern whenever J_κ is of type-2.

```
Subroutine Find-Good-Split_k(f, ε, δ, c, ξ).

Input: Query access to a function $f\colon I \to \mathbb{R}\cup\{*\}$, parameters $\varepsilon,\delta \in (0,1)$, and $c \in [k-1]$.
We let $c_1 > 1$ be a large enough (absolute) constant.
Output: a sequence $i_1 < \ldots < i_k$ with $f(i_1) < \ldots < f(i_k)$, or fail.

  1. Repeat the following procedure $t = c_1 k/(\varepsilon\xi^2)\cdot\log(1/\delta)$ times:
     (a) Sample $\boldsymbol{w},\boldsymbol{z} \sim I$, and consider the functions $f_{\boldsymbol{z},\boldsymbol{w}}\colon I\cap(-\infty,\boldsymbol{z}) \to \mathbb{R}\cup\{*\}$ and
         $f'_{\boldsymbol{z},\boldsymbol{w}}\colon I\cap[\boldsymbol{z},\infty) \to \mathbb{R}\cup\{*\}$ given by
         $f_{\boldsymbol{z},\boldsymbol{w}}(i) = \begin{cases} f(i) & f(i) < f(\boldsymbol{w}) \\ * & \text{o.w.} \end{cases}$ and $f'_{\boldsymbol{z},\boldsymbol{w}}(i) = \begin{cases} f(i) & f(i) \geq f(\boldsymbol{w}) \\ * & \text{o.w.} \end{cases}$.   (4.4)
     (b) Run Find-Monotone_c($f_{\boldsymbol{z},\boldsymbol{w}}, \varepsilon\xi/3, \delta/3$) and Find-Monotone_{k-c}($f'_{\boldsymbol{z},\boldsymbol{w}}, \varepsilon\xi/3, \delta/3$).
  2. If both runs of Line 1b are successful for some iteration and some $\boldsymbol{w}$, $\boldsymbol{z}$ and $c$, then we
     output the combination of their outputs which forms a length-$k$ increasing subsequence
     of $f$; otherwise, output fail.
```

Figure 4.5: Description of the `Find-Good-Split` subroutine.

Notice that if J_1 is of type-2, the $(12\ldots(k-1))$-pattern returned in Line 3 can be combined with x to form a $(12\ldots k)$-pattern. Hence, we may assume that J_1 is of type-1. Furthermore, if J_{k-2} is of type-1, the $(12\ldots(k-1))$-pattern found in Line 2 can be combined with y to form a $(12\ldots k)$-pattern, and hence, we may assume that J_{k-2} is of type-2. Thus, there exists some $\kappa \in [k-3]$ where J_κ is of type-1 and $J_{\kappa+1}$ is of type-2. Since J_κ comes before $J_{\kappa+1}$, and since non-$*$ elements in f_κ lie below the non-$*$ elements of $f'_{\kappa+1}$, we can combine the $(12\ldots(\kappa+1))$-pattern in f_κ with the $(12\ldots(k-\kappa-1))$-pattern in $f'_{\kappa+1}$. $\square$

Handling the Fitting Case: The `Find-Good-Split` Sub-Routine

In this section, we describe the `Find-Good-Split` subroutine, which corresponds to the fitting case from Section 4.1.1.

Lemma 4.11. *Consider the randomized algorithm* `Find-Good-Split`$_k(f,\varepsilon,\delta,c,\xi)$*, described in Figure 4.5, which takes as input five parameters:*

- *Query access to a function $f\colon I \to \mathbb{R}\cup\{*\}$,*
- *Two parameters $\varepsilon,\delta \in (0,1)$,*

- *An integer $c \in [k-1]$, and*
- *A parameter $\xi \in (0,1]$,*

and outputs either a length-k increasing subsequence or fail*.*

Suppose that there exists an interval-tuple pair (I', T) which is $(c, 1/(6k), \varepsilon)$-splittable and $|I'|/|I| \geq \xi$. Then, the algorithms `Find-Good-Split`$_k(f, \varepsilon, \delta, c, \xi)$ *finds a $(12\dots k)$-pattern of f with probability $1-\delta$.*

Proof. Let (I', T) be $(c, 1/(6k), \varepsilon)$-splittable, and let L, M, R be the contiguous intervals splitting I' as in Definition 3.7. Furthermore, let $T^{(L)}$ and $T^{(R)}$ be as in Definition 3.7. Writing

$$m_1 = \text{rank}\left(\left\{f(i_c) : (i_1, \dots, i_c) \in T^{(L)}\right\}, |T|/3\right),$$
$$m_2 = \text{rank}\left(\left\{f(i_c) : (i_1, \dots, i_c) \in T^{(L)}\right\}, 2|T|/3\right),$$

as the $(|T|/3)$-largest and $(2|T|/3)$-largest elements in $\left\{f(i_c) : (i_1, \dots, i_c) \in T^{(L)}\right\}$ (taking multiplicity into account). Let $T_l^{(L)}$ be the $(12\dots c)$-patterns in $T^{(L)}$ where the c-th index is at most m_1, and $T_h^{(R)}$ be the $(k-c)$-patterns in $T^{(R)}$ whose $(c+1)$-th index is larger than m_2. Notice that $|T_l^{(L)}|, |T_h^{(R)}| \geq |T|/3$, and that any $(12\dots c)$-pattern from $T_l^{(L)}$ can be combined with any $(12\dots(k-c))$-pattern from $T_h^{(R)}$ to form a $(12\dots k)$-pattern. Furthermore, there exists at least $|T|/3$ indices in I' whose function value lies in $[m_1, m_2]$.

Consider the event, defined over the randomness of $\boldsymbol{w}, \boldsymbol{z} \sim I$ that: $\boldsymbol{z} \in M$; and $\boldsymbol{w}$ satisfies $f(\boldsymbol{w}) \in [m_1, m_2]$. This event occurs at some iteration of Line 1, with probability at least $1 - \delta/3$; this is because there are $|M| \geq |I'|/(6k) \geq (\xi/(6k))|I|$ valid indices for $\boldsymbol{z}$, and there are at least $|T|/3 \geq (\varepsilon/3)|I'| \geq (\varepsilon\xi/3)|I|$ indices for $\boldsymbol{w}$, so the probability that the pair $(\boldsymbol{z}, \boldsymbol{w})$ satisfies the requirements is at least $\varepsilon\xi^2/(18k)$. We obtain the desired bound by the setting of t, since c_1 is set to a large enough constant.

Notice that when this event occurs, the $(12\dots c)$-patterns in $T_l^{(L)}$ all lie in $f_{\boldsymbol{z},\boldsymbol{w}}$, and the $(12\dots(k-c))$-patterns in $T_h^{(R)}$ all lie in $f'_{\boldsymbol{z},\boldsymbol{w}}$. In particular, $f_{\boldsymbol{z},\boldsymbol{w}}$ contains at least $|T|/3 \geq (\varepsilon/3)|I'| \geq (\varepsilon\xi/3)|I|$ disjoint $(12\dots c)$-patterns, and $f'_{\boldsymbol{z},\boldsymbol{w}}$ similarly contains at least $(\varepsilon\xi/3)|I|$ disjoint $(12\dots(k-c))$-patterns. Thus, by the inductive hypothesis, Line 1b finds a $(12\dots c)$-pattern in $f_{\boldsymbol{z},\boldsymbol{w}}$ and a $(12\dots(k-c))$-pattern in $f'_{\boldsymbol{z},\boldsymbol{w}}$ with probability at least $1 - 2\delta/3$, and these can be combined to give a $(12\dots k)$-pattern of f. □

The Main Algorithm

Consider the description of the main algorithm in Figure 4.6. We prove Theorem 4.7 by induction on k. The proof uses Lemma 4.9, Lemma 4.10, and Lemma 4.11.

Proof of Theorem 4.7.

Subroutine $\texttt{Find-Monotone}_k(f, \varepsilon, \delta)$.

Input: Query access to a function $f\colon I \to \mathbb{R} \cup \{*\}$, parameters $\varepsilon, \delta \in (0,1)$. We let $c_1, c_2, c_3 > 0$ be large enough constants, and let $p = P(k \log(1/\varepsilon))$, where $P\colon \mathbb{R} \to \mathbb{R}$ is a polynomial of large enough (constant) degree.

Output: a sequence $i_1 < \ldots < i_k$ with $f(i_1) < \ldots < f(i_k)$, or fail.

```
1. Run $\texttt{Sample-Suffix}_k(f, \varepsilon/p, \delta)$.
2. Repeat the following for $c_1 \log(1/\delta) \cdot p \cdot k^5/\varepsilon^2$ many iterations:
   (a) Sample $\boldsymbol{x} \sim I$ uniformly at random. If $f(\boldsymbol{x}) = *$, proceed to the next iteration.
       Otherwise, if $k = 1$ output $\boldsymbol{x}$ and proceed to Step 3, and if $k \geq 2$ proceed to the next step.
   (b) For each $t \in [\log n]$, sample $\boldsymbol{y}_t \sim [\boldsymbol{x} + 2^t/(12k), \boldsymbol{x} + 2^t]$ uniformly at random. If
       there exists at least one $t$ where $f(\boldsymbol{y}_t) > f(\boldsymbol{x})$, set
           $$\boldsymbol{y} = \max\{\boldsymbol{y}_t : t \in [\log n] \text{ and } f(\boldsymbol{y}_t) > f(\boldsymbol{x})\}, \qquad (4.5)$$
       let $t^* \in [\log n]$ be the index for which $\boldsymbol{y}_{t^*} = \boldsymbol{y}$, and continue to the next line.
       Otherwise, i.e. if $f(\boldsymbol{y}_t) \not> f(\boldsymbol{x})$ for every $t$, continue to the next iteration.
   (c) If $k = 2$, output $(\boldsymbol{x}, \boldsymbol{y})$ and proceed to Step 3. If $k > 2$, continue to the next line.
   (d) Here $k \geq 3$. Set $\ell = 4p/\varepsilon$ and perform the following.
       i. Consider the collection $\mathcal{J}$ of $k-2$ intervals $J_1, \ldots, J_{k-2}$ appearing in order
          within $[\boldsymbol{x}, \boldsymbol{y}]$, given by setting, for every $i \in [k-2]$,
              $$J_i = \left[\boldsymbol{x} + \frac{2^{t^*}}{12k} \cdot \ell^{-(k-1-i)}, \boldsymbol{x} + \frac{2^{t^*}}{12k} \cdot \ell^{-(k-2-i)}\right), \qquad (4.6)$$
          and run $\texttt{Find-Within-Interval}_k(f, \varepsilon/2p, \delta/2, \boldsymbol{x}, \boldsymbol{y}, \mathcal{J})$.
       ii. For each $t' \in [t^* - 3k \log \ell, t^*]$ do the following.
          Consider the interval $J_{t'} = [\boldsymbol{x} - 2^{t'}, \boldsymbol{x} + 2^{t'}]$, and the restricted function
          $g_{t'}\colon J_{t'} \to \mathbb{R} \cup \{*\}$ given by $g_{t'} = f|_{J_{t'}}$. For every $c_0 \in [k-1]$, run
          $\texttt{Find-Good-Split}_k(g_{t'}, \varepsilon/(c_2 k^5), \delta/2, c_0, 1/4)$.
3. If a length-$k$ monotone subsequence of $f$ is found, output it. Otherwise, output fail.
```

Figure 4.6: Description of the $\texttt{Find-Monotone}_k$ subroutine.

Base Case: $k = 1$.

Recall that f has at least $\varepsilon|I|$ non-$*$ values. Thus, with probability at least $1-\delta$, a non-$*$ value is observed after sampling $\boldsymbol{x} \sim I$ at least $(1/\varepsilon) \cdot \log(1/\delta)$ times. It follows that with probability at least $1-\delta$, Line 2a of our main algorithm, given in Figure 4.6, samples $\boldsymbol{x} \neq *$ in one of its iterations.

Inductive Step: proof of Theorem 4.7 for $k \geq 2$, under the assumption that it holds for every k' with $1 \leq k' < k$.

Let $p = P(k \log(1/\varepsilon))$ (recall that $P(\cdot)$ is a polynomial of sufficiently large (constant) degree). Apply Theorem 4.6 to f.

Suppose, first, that (1) of Theorem 4.6 holds. So, there exists a set $H \subset [n]$ of indices that start an $(\alpha, Ck\alpha)$-growing suffix, with $\alpha|H| \geq (\varepsilon/p)n$, for some $\alpha \in (0,1)$. By Lemma 4.9, the call for $\texttt{Sample-Suffix}_k(f, \varepsilon/p, \delta)$ in Line 1 outputs a length-k monotone subsequence of f with probability at least $1-\delta$. Now suppose that (2) of Theorem 4.6 holds, and let $(I_1, T_1), \ldots, (I_s, T_s)$ be a $(c, 1/(6k), \alpha)$-splittable collection for some $\alpha \geq \Omega(\varepsilon/k^5)$ and $c \in [k-1]$, satisfying (4.2) and, moreover, that any $J \subset I$ with $J \supset I_h$ for some $h \in [s]$ contains $(\varepsilon/p)|J|$ disjoint $(12 \ldots k)$-patterns. Let `Event` be the event that, for a particular iteration of Lines 2a and 2b, $\boldsymbol{x}$ is the 1-entry of some k-tuple from T_h, for some $h \in [s]$, and $\boldsymbol{y}_t$ is the $(c+1)$-entry of some (possibly other) k-tuple in T_h, where t is such that $|I_h| \leq 2^t < 2|I_h|$.

Claim 4.12. $\mathbf{Pr}[\texttt{Event}] \geq \varepsilon\alpha/(2p)$.

Proof. For each $h \in [s]$, let A_h and B_h be the collections of 1- and $(c+1)$-entries of patterns in T_h. Then

$$\sum_{h=1}^{s} |A_h| = \sum_{h=1}^{s} |T_h| \geq \alpha \sum_{h=1}^{s} |I_h| \geq \frac{\varepsilon}{p} \cdot |I|.$$

The first inequality follows from the assumption that (I_h, T_h) is $(c, 1/(6k), \alpha)$-splittable, and the second inequality follows from the assumption that (4.2) holds.

As a result, the probability over the draw of $\boldsymbol{x} \sim I$ in Line 2a that $\boldsymbol{x} \in A_h$ is at least ε/p. Fix such an $\boldsymbol{x}$, and consider $t \in [\log n]$ for which $|I_h| \leq 2^t < 2|I_h|$. Notice that $B_h \subset [\boldsymbol{x} + 2^t/(12k), \boldsymbol{x} + 2^t]$ since $2^{t-1} \leq |I_h| < 2^t$, and that the distance between any index of A_h and B_h is at least $|I_h|/(6k) \geq 2^t/(12k)$ since (I_h, T_h) is $(c, 1/(6k), \alpha)$-splittable. Therefore, the probability over the draw of $\boldsymbol{y}_t \sim [\boldsymbol{x} + 2^t/(12k), \boldsymbol{x} + 2^t]$ that $\boldsymbol{y}_t \in B_h$ is at least $|B_h|/2^t \geq |T_h|/(2|I_h|) \geq \alpha/2$. □

By the previous claim, since we have $c_1 \cdot \log(1/\delta) \cdot p \cdot k^5/\varepsilon^2$ iterations of Lines 2a and 2b, with probability at least $1-\delta/2$, `Event` holds in some iteration (using the lower bound $\alpha \geq \Omega(\varepsilon/k^5)$ and the choice of c_1 as a large constant). Consider the first execution of Line 2a and Line 2b where `Event` holds (assuming such an execution exists). Let $h \in [s]$ and $t \in [\log n]$ be the corresponding parameters, i.e., h and t are set so $\boldsymbol{x}$ is the first index of a k-tuple in T_h, $\boldsymbol{y}_t$ is the $(c+1)$-th index in another k-tuple in T_h, and $|I_h| \leq 2^t < 2|I_h|$. We consider this iteration of Line 2, and assume that `Event` holds with these parameters for the rest of the proof. Notice that $\boldsymbol{y}$, as defined in (4.5), satisfies $\boldsymbol{y} \geq \boldsymbol{y}_t$ (as $f(\boldsymbol{y}) > f(\boldsymbol{x})$) and hence $t^* \geq t$.

Note that if $k = 2$, the pair $(\boldsymbol{x}, \boldsymbol{y})$, which is a (12)-pattern in f, is output in Line 2c, so the proof is complete in this case. From now on, we assume that $k \geq 3$. We break up the analysis into two cases: $t^* \geq t + 3k \log \ell$ and $t^* < t + 3k \log \ell$. Suppose first that $t^* \geq t + 3k \log \ell$. We now observe a few facts about the collection $\mathcal{J}$ specified in (4.6). First, notice that $J_1, \dots, J_{k-2}$ appear in order from left-to-right, and they lie in $[\boldsymbol{x}, \boldsymbol{y}]$ (as $\boldsymbol{y} = \boldsymbol{y}_{t^*} \in [\boldsymbol{x} + 2^{t^*}/(12k), 2^{t^*}]$). Second, in the next claim we show that for every $i \in [k-2]$, the interval J_i contains $(\varepsilon/2p)|J_i|$ disjoint $(12 \dots k)$-patterns.

Claim 4.13. *J_i contains $(\varepsilon/2p)|J_i|$ disjoint $(12 \dots k)$-patterns.*

Proof. Let J'_i be the interval given by

$$J'_i = I_h \cup \left[\boldsymbol{x}, \boldsymbol{x} + \frac{2^{t^*}}{12k} \cdot \ell^{-(k-2-i)}\right].$$

Observe that

$$|J'_i \setminus J_i| \leq 2^t + \frac{2^{t^*}}{12k} \cdot \ell^{-(k-1-i)} \leq \frac{2^{t^*}}{6k} \cdot \ell^{-(k-1-i)} = \frac{2}{\ell} \cdot \frac{2^{t^*}}{12k} \cdot \ell^{-(k-2-i)} \geq \frac{2}{\ell} \cdot |J'_i| = \frac{\varepsilon}{2p} \cdot |J'_i|,$$

where for the second inequality we used the bound $t^* - t \geq 3k \log \ell \geq \log(12) + \log k + (k-2) \log \ell$, and that $\ell = 4p/\varepsilon$.

We have by Theorem 4.6, that J'_i contains at least $(\varepsilon/p)|J'_i|$ disjoint $(12 \dots k)$-patterns in f. Hence, the number of disjoint $(12 \dots k)$-patterns in J_i is at least:

$$\frac{\varepsilon}{p} \cdot |J'_i| - |J'_i \setminus J_i| \geq \frac{\varepsilon}{2p} \cdot |J'_i| \geq \frac{\varepsilon}{2p} \cdot |J_i|,$$

as required. □

By Lemma 4.10, Line 2(d)i outputs a $(12 \dots k)$-pattern in f with probability at least $1 - \delta/2$. By a union bound, we obtain the desired result.

Suppose, on the other hand, that $t^* \leq t + 3k \log \ell$. In this case, as $2^{t-1} \leq |I_h| \leq 2^{t^*}$ (by choice of t), for one of the values of t' considered in Line 2(d)ii we have $2^{t'-1} \leq |I_h| < 2^{t'}$; fix this t'. The interval $J_{t'}$, defined in Line 2(d)ii, hence satisfies $|I_h|/|J_{t'}| \geq 1/4$. As a result, and since $I_h \subset J_{t'}$ (because $t \leq t^*$), the function $g \colon J \to \mathbb{R} \cup \{*\}$ contains an interval-tuple pair (I_h, T_h) which is $(c, 1/(6k), \alpha)$-splittable. By Lemma 4.11, once Line 2(d)ii considers $c_0 = c$, the sub-routine $\texttt{Find-Good-Split}_k(g, \varepsilon/(c_2 k^5), \delta/2, c, 1/4)$ will output a $(12 \dots k)$-pattern of $g_{t'}$ (which is also a $(12 \dots k)$-pattern of f) with probability at least $1 - \delta/2$. Hence, we obtain the result by a union bound. □

Query Complexity and Running Time

It remains to prove Lemma 4.8, estimating the number of queries made by `Find-Monotone`, as well as its total running time.

Proof of Lemma 4.8. We first claim that the running time is bounded by an expression of the form poly(k) times the query complexity of `Find-Monotone`, where the poly$(\cdot)$ term is of constant degree. Indeed, the only costly operations (in terms of running time) other than querying that our algorithm conducts involve:

- Determining whether the value of f at a certain point is $*$ or not; to this end, note that for any f we need to evaluate along the way, $f(x)$ is marked by $*$ if and only if it does not belong to some interval in $\mathbb{R}$, whose endpoints are determined by the recursive calls that led to it. Since the recursive depth is at most k, this means that the complexity of the above operation is $O(k)$.[4]

- Given an ordered set of queried elements Q at some point along the algorithm, determining whether these elements contain a c-increasing subsequence for $c \leq k$ (this action is taken, e.g., in the last part of `Find-Monotone`). This operation can be implemented in time $O(c|Q|)$. Now, the number of such operations that each queried element participates in is at most k,[5] and a simple double counting argument implies that the running time of these operations altogether is at most $O(k^2)$ times the total query complexity.

It remains now to prove the bound on the query complexity. Recall that $P\colon \mathbb{R} \to \mathbb{R}$ is a fixed polynomial; write $p_{k,\varepsilon} = P(k \log(1/\varepsilon))$. We fix n, which upper bounds the length of all intervals defining input functions. Let $\Phi(k, \varepsilon, \delta)$ be the maximum number of queries made by $\texttt{Find-Monotone}_k(f, \varepsilon, \delta)$. Let

$$\begin{aligned}
\Phi^{(1)}(k,\varepsilon,\delta) &= \text{ query complexity of } \texttt{Sample-Suffix}_k(f,\varepsilon,\delta).\\
\Phi^{(2)}(k,\varepsilon,\delta) &= \begin{array}{l}\text{query complexity of } \texttt{Find-Within-Interval}_k(f,\varepsilon,\delta,x,y,\mathcal{J}),\\ \text{where } |\mathcal{J}| = k-2.\end{array}\\
\Phi^{(3)}(k,\varepsilon,\delta,\xi) &= \begin{array}{l}\text{query complexity of } \texttt{Find-Good-Split}_k(f,\varepsilon,\delta,c,\xi),\\ \text{where } c \in [k-1].\end{array}
\end{aligned}$$

[4] In fact, this complexity can be improved to $O(1)$ if, instead of working with functions of the form $f\colon I \to \mathbb{R} \cup \{*\}$, we would have worked with function $f\colon I \to \mathbb{R}$ and received the interval of "non-$*$ values" as an input to the recursive call.

[5] More precisely, for the purpose of this section, if an element is queried $t > 1$ times by our algorithm then we think of it as contributing t to the total query complexity (since our goal is to prove upper bounds here – not lower bounds – this perspective is clearly valid); and in this case, the number of operations as above in which it participates is at most $k \cdot t$.

By Lemma 4.9, as well as an inspection of Figure 4.4 and Figure 4.5, we have:

$$\begin{aligned}
\Phi^{(1)}(k,\varepsilon,\delta) &\le p_{k,\varepsilon}\cdot\frac{1}{\varepsilon}\cdot\log(1/\delta)\cdot\log n\\
\Phi^{(2)}(k,\varepsilon,\delta) &\le 2k\cdot\Phi(k-1,\varepsilon/2,\delta/(2k))\\
\Phi^{(3)}(k,\varepsilon,\delta,\xi) &\le \frac{c_1k\log(1/\delta)}{\varepsilon\xi^2}\cdot\Phi(k-1,\varepsilon\xi/3,\delta/3).
\end{aligned}$$

Lastly, inspecting Figure 4.6, we have

$$\begin{aligned}
\Phi(k,\varepsilon,\delta) \le{}& \Phi^{(1)}(k,\varepsilon/p_{k,\varepsilon},\delta)+\\
&c_1p_{k,\varepsilon}\frac{k^5}{\varepsilon^2}\log(1/\delta)\Big(1+\log n+\Phi^{(2)}\left(k,\varepsilon/(2p_{k,\varepsilon}),\delta/2\right)+\Phi^{(3)}\left(k,\varepsilon/(c_2k^5),\delta/2,1/4\right)\Big)\\
\le{}& q_{k,\varepsilon}\cdot\frac{1}{\varepsilon^2}\cdot\log(1/\delta)\cdot\log n\,+q_{k,\varepsilon}\cdot\frac{1}{\varepsilon^3}\cdot(\log(1/\delta))^2\cdot\Phi(k-1,\varepsilon/q_{k,\varepsilon},\delta/(3k))\\
\le{}& \left(k^k\cdot(\log(1/\varepsilon))^k\cdot\frac{1}{\varepsilon}\cdot\log(1/\delta)\right)^{O(k)}\cdot\log n,
\end{aligned}$$

where $Q\colon\mathbb{R}\to\mathbb{R}$ is a fixed polynomial of large enough (constant) degree and $q_{k,\varepsilon}=Q(k\log(1/\varepsilon))$. For the last line we use that $\Phi^{(2)}(1,\cdot,\cdot)=\Phi^{(2)}(2,\cdot,\cdot)=\Phi^{(3)}(1,\cdot,\cdot,\cdot)=\Phi^{(3)}(2,\cdot,\cdot,\cdot)=0$, and we note that the parameter replacing ε never falls below $\varepsilon/(k\log(1/\varepsilon))^{O(k)}$, so the factor of $\log n$ at each iteration is at most $\left(k^k(\log(1/\varepsilon))^k(1/\varepsilon)\log(1/\delta)\right)^{O(k)}$. □

Chapter 5

General Patterns: Stitching, Lower Bounds, and Hierarchies

The results in this chapter appear in [23].

5.1 Introduction

In this chapter, we continue the discussion from Chapters 3 and 4 on one-sided error testing for forbidden order patterns in sequences $f\colon [n] \to \mathbb{R}$. Again, the *forbidden order pattern* is a permutation $\pi = (\pi_1, \dots, \pi_k)$ of $[k]$, viewed here as a sequence of length k. Unlike the monotone case, here we do not pose any restriction on the permutation discussed. We are interested in testing for π-freeness, i.e., of not containing an order-isomorphic copy of π as a subsequence. Here, two sequences $x = (x_1, \dots, x_k)$ and $y = (y_1, \dots, y_k)$ are *order-isomorphic* if for any $i \neq j$, $x_i < x_j$ holds if and only if $y_i < y_j$. That is, if the relative order of the elements in both sequences is the same. In other words, f contains π if there exist k integers $i_1 < \cdots < i_k \in [n]$ such that $f(i_a) < f(i_b)$ if and only if $\pi(a) < \pi(b)$. Accordingly, f is *π-free* if it does not contain the pattern π.

The focus here is on query complexity; all of our tests run in time *linear* in the number of queries they make. This is because they work by checking whether the queried subsequence contains the forbidden pattern. But this in turn can be performed efficiently, building on an algorithm of Guillemot and Marx [90] which determines whether a given sequence contains a fixed pattern in time that is linear in the size of the sequence.

Distance function All results are stated here for the Hamming distance function, which is most standard in property testing, but they also hold for the stronger *deletion distance*, defined as follows: $\mathrm{dist}_{\mathrm{del}}(f, g)$ is the minimal number of value modifications, deletions, and insertions needed to turn f into g. This follows from the fact that the Hamming distance and the deletion distance of a sequence f to π-freeness are always equal: Indeed, if S is a set of entries of a function $f\colon [n] \to \mathbb{R}$

whose deletion turns f into a π-free sequence, then it is possible to turn f into a π-free sequence using $|S|$ value modifications by initializing $T = S$ and iteratively applying the following until T is empty: Find an $x \in T$ with a neighboring entry $y \notin T$, set $f(x) = f(y)$, and remove x from T. This way, if f restricted to $[n] \setminus S$ is π-free, then so is f after these value modifications.

In particular, this implies that the distance of a sequence f to π-freeness is closely related to the maximum size of a set $\mathcal{C}$ of pairwise-disjoint π-copies in f: On one hand, if f is ε-far from π-freeness then we cannot delete all π-copies in $\mathcal{C}$ with less than εn entry deletions, so $|\mathcal{C}| \geq \varepsilon n/|\pi|$. On the other hand, if $|\mathcal{C}| \geq \varepsilon n$ then trivially f is ε-far from π-freeness.

Organization of the results Below we present the results of Newman et al. [109] on the problem of testing π-freeness. We then provide our results in Section 5.1.2. Our results stated here are in the non-adaptive case, and seem to yield a relatively good general understanding of this case. All results in [109] only consider one-sided testing, and we also follow this paradigm. An additional discussion on a hierarchy of adaptivity in this problem can be found in the full version of the results presented here [23].

5.1.1 Previous Work

We describe here the previous state of knowledge on testing pattern-freeness in the non-monotone case; all results here are established in [109], and focus on one-sided tests. First, any pattern of length k has a non-adaptive one-sided test making $O(\varepsilon^{-1/k} n^{1-1/k})$ queries. This is the sample-based test, that samples a uniformly random set of elements in the input sequence of the required size and accepts the input (i.e., indicates that it is π-free) if the queried subsequence is π-free. In what follows, this test is called the *sampler*.

The second line of results concerns patterns of length 3. Due to symmetry considerations, it is enough to consider the pattern $\pi = (1, 3, 2)$. For this choice of π, it is shown that:

- There is an *adaptive* one-sided ε-test for π-freeness making $(\varepsilon^{-1} \log n)^{O(1)}$ queries.
- Any *non-adaptive* 1/9-test for π-freeness has query complexity $\Omega(\sqrt{n})$ – an exponential separation from the adaptive case! It is interesting to note that while the lower bound in [109] was only obtained for one-sided tests, a similar lower bound for two-sided tests may be derived using similar (yet more technical) ideas.
- There is a *non-adaptive* one-sided ε-test for π-freeness making $\sqrt{n}(\varepsilon^{-1} \log n)^{O(1)}$ queries. Thus, the non-adaptive bounds for $\pi = (1, 3, 2)$ are tight up to an $(\varepsilon^{-1} \log n)^{O(1)}$ factor.

The $\Omega(\sqrt{n})$ non-adaptive lower bound from Section 5.1.1 actually applies to *any* non-monotone pattern. Moreover, this bound can be strengthened for certain patterns: For any odd k, any one-sided non-adaptive test for the pattern $\pi = (1, k, k-1, 2, 3, k-2, \ldots, (k+1)/2)$ requires $\Omega(n^{1-2/(k+1)})$ queries.

Discussion on previous results The results in [109] and in the previous two chapters essentially settle two special cases: The monotone patterns of any length, and the patterns of length 3. However, the general task of understanding the query complexity of optimal tests for π-freeness – for any π – both in the adaptive and the non-adaptive case, has remained wide open. The major open problems that Newman et al. proposed are the following.

Adaptive case Is it true that π-freeness is testable adaptively with query complexity polylogarithmic in n for *any* pattern π?

Non-adaptive case How does the structure of a pattern π correlate with the query complexity of an optimal (one-sided) non-adaptive test for π-freeness? In particular, do there exist infinitely many patterns π for which π-freeness is testable with query complexity that is $O(n^{0.99})$?

5.1.2 Our Contributions

In this chapter, we address the non-adaptive case, achieving good (though not yet complete) understanding of this case. Along the way, we discover many interesting and surprising phenomena. The details are presented below. We remark that additional results exploring how partial adaptivity helps in the problem of testing π-freeness, in particular for the pattern $\pi = (1, 3, 2)$, appear in [23].

Our first main result is an improved general upper bound that holds for all patterns.

Theorem 5.1. *For any pattern π of length $k \geq 3$, π-freeness has a one-sided non-adaptive ε-test whose query complexity is $O(\varepsilon^{-\frac{1}{k-1}} n^{1-\frac{1}{k-1}})$.*

This bound improves upon all previously known upper bounds for non-monotone patterns, as the query complexity it suggests is better than both the sample-based upper bound and the upper bound for patterns of length 3, as appeared Section 5.1.1.

At first glance, an upper bound of $O(\varepsilon^{-\frac{1}{k-1}} n^{1-\frac{1}{k-1}})$ seems to only be a slight improvement over the $O(\varepsilon^{-\frac{1}{k}} n^{1-\frac{1}{k}})$ sample-based upper bound. However, quite surprisingly, this upper bound is tight in both n and ε for any $k \geq 3$. In other words, the optimal non-adaptive one-sided test for some patterns is only slightly more query-efficient than the sampler!

Theorem 5.2. *Let π be a pattern of length $k \geq 3$, and suppose that $|\pi^{-1}(1) - \pi^{-1}(k)| = 1$. Then the query complexity of any non-adaptive one-sided ε-test for π-freeness is $\Omega\left(\varepsilon^{-\frac{1}{k-1}} n^{1-\frac{1}{k-1}}\right)$.*

This improves the non-adaptive lower bounds for any pattern of this type, whose length is at least four. For the non-monotone patterns of length 3, this results determines the correct dependence in ε.

The combination of Theorem 5.2 with the results in Section 5.1.1 demonstrates a surprising phenomenon: While the deletion distance between the patterns $\pi_1 = (1, 2, \ldots, k)$ and $\pi_2 = (k, 1, 2, \ldots, k-1)$ is only 2, the query complexity of non-adaptive one-sided testing for π_1-freeness differs significantly from that of π_2-freeness. For π_1-freeness this query complexity is polylogarithmic in n, and so π_1 is the easiest to test among patterns of length k, while π_2-freeness has a query complexity of $\Theta\left(\varepsilon^{-\frac{1}{k-1}} n^{1-\frac{1}{k-1}}\right)$, making π_2 one of those patterns that are hardest to test with non-adaptive one-sided tests. The proofs of Theorems 5.1 and 5.2 appear in Sections 5.2 and 5.3, respectively.

The next lower bound is perhaps even more surprising. It provides (along with Theorem 5.1) an almost tight bound on the query complexity of an optimal non-adaptive one-sided test for almost all patterns, implying that this query complexity is usually only marginally better than that of the most basic sample-based test.

Theorem 5.3. *Let π be picked uniformly at random from all patterns of length k. The following holds with probability $1-o(1)$ (where the $o(1)$ term tends to zero as $k \to \infty$): The query complexity of any non-adaptive one-sided ε-test for π-freeness is $\Omega\left(\varepsilon^{-\frac{1}{k-3}} n^{1-\frac{1}{k-3}}\right)$.*

Both Theorems 5.2 and 5.3 are actually special cases of a general pattern-dependent lower bound that we establish. This lower bound applies to any pattern, and depends heavily on the structure of the pattern. We believe that this lower bound is tight (up to polylogarithmic factors) for any pattern. Interestingly, it is not clear how to describe the lower bound in a compact closed form, but given a pattern π of length k, the corresponding bound can be computed in constant time (that depends only on k). Later, as an important special case of this strong yet hard-to-digest bound, we provide a slightly weaker pattern-dependent lower bound that has a more natural combinatorial characterization, and is therefore easier to analyze. See Theorem 5.8 and the resulting Corollaries 5.9 and 5.10 for more details.

In order to describe our general lower bound, we shall first provide some definitions.

Definition 5.4. *Let $\pi = (\pi_1, \ldots, \pi_k)$ be a pattern of length k. A subsequence σ of π is* consecutive *if $\sigma = (\pi_i, \ldots, \pi_j)$ for some $1 \le i \le j \le k$; in this case we write $\sigma = \pi[i,j]$.*

A partition *$\Lambda = (\sigma_1, \ldots, \sigma_\ell)$ of the pattern π consists of consecutive subsequences $\sigma_1 = \pi[1, r_1], \sigma_2 = \pi[r_1+1, r_2], \ldots, \sigma_\ell = \pi[r_{\ell-1}+1, k]$, and its* size *is $|\Lambda| = \ell$.*

A signed partition *$P = (\Lambda, S)$ of the pattern π consists of a partition Λ as above, and a sign vector $S = (s_1, \ldots, s_\ell) \in \{+,-\}^\ell$. For any σ_i of length bigger than one, the corresponding sign s_i must satisfy the following. If $\min \sigma_i$ appears before $\max \sigma_i$ in π, then the direction sign of σ_i is $-$, and otherwise, the direction sign is $+$. The* size *of P is $|\Lambda| = |S| = \ell$.*

Let P be a signed partition as above. Define $r_0 = 0$, and for any $1 \le i \le \ell$, denote the length of σ_i by k_i (so $\sum_{i=1}^{\ell} k_i = k$). Consider the sequence $f_P \colon [k^2] \to \mathbb{R}$ defined as follows. For any $1 \le j \le k_i$ and $0 \le m \le k-1$, we take $f_P(r_i k + m k_i + j) = m + \pi_{r_i+j}/2k$ for any $0 \le i \le \ell - 1$ where s_i is $+$, and $f_P(r_i k + m k_i + j) = (k-1-m) + \pi_{r_i+j}/2k$ for any $\le i \le \ell - 1$ where s_i is a $-$.

Note that for any $0 \le m \le k-1$, the set of all entries $x \in [k^2]$ satisfying $m < f_P(x) < m+1$ is a π-copy. We say that such a π-copy is trivial. *We say that P is* unique *if f_P does not contain* *non-trivial π-copies, and denote by $\mathcal{U}(\pi)$ the set of all unique signed partitions of π. Finally, the* unique signed partition number *(USPN) of π is $u(\pi) = \max_{P \in \mathcal{U}(\pi)} |P|$.*

Our lower bound for testing π-freeness is closely related to the USPN of π.

Theorem 5.5. *Let π be any pattern. Any non-adaptive one-sided ε-test for π-freeness has query complexity $\Omega\left(\varepsilon^{-1/u(\pi)} n^{1-1/u(\pi)}\right)$.*

The USPN of a pattern obviously depends only on the pattern (and not on the input sequence size), so it can be computed in constant time, that depends only on k. Thus, given a pattern π and parameters n, ε, one can compute the lower bound obtained from Theorem 5.5 in constant time.

The proof of Theorem 5.2 follows from Theorem 5.5 by showing that for any pattern π of length k which satisfies $|\pi^{-1}(1) - \pi^{-1}(k)| = 1$, it holds that $u(\pi) = k - 1$; actually these are the only patterns of length k whose USPN is $k - 1$, and no pattern of length $k > 1$ has USPN that equals k, as can be derived from results that are discussed later.

We conjecture that the lower bound of Theorem 5.5 is tight up to a multiplicative term that is polynomial in ε and $\log n$. That is, we conjecture that the USPN, $u(\pi)$, is the correct parameter of π that determines how hard it is to non-adaptively test π-freeness using one-sided tests.

Conjecture 5.6. *For any pattern π of any fixed length, π-freeness has a non-adaptive one-sided ε-test making $\tilde{\Theta}_\varepsilon\left(n^{1-1/u(\pi)}\right)$ queries.*

A multiplicative term of $\log n$ is necessary to make Conjecture 5.6 hold for monotone patterns π (for which $u(\pi) = 1$), since there is a lower bound of $\Omega(\log n)$ for testing monotonicity [66], that can be generalized to testing π-freeness for any pattern π of length at least 2. For non-monotone patterns, an even stronger conjecture can be given, namely that the number of queries required by a non-adaptive one-sided ε-test is $\Theta_\varepsilon(n^{1-1/u(\pi)})$ (without the polylogarithmic term in n).

Combinatorial characterizations related to the general lower bound Motivated by Theorem 5.5, it is desirable to find natural necessary and sufficient combinatorial conditions for uniqueness of a signed partition of a given pattern π. Our next main result provides a useful sufficient condition. For the result, we need some more definitions.

Definition 5.7. *Let $\sigma = \pi[x, y]$ and $\sigma' = \pi[x', y']$ of π be disjoint consecutive subsequences of length at least two, and let $x' \le m, M \le y'$ be the indices satisfying $\pi_m = \min \pi[x', y']$ and $\pi_M = \max \pi[x', y']$. We say that σ' is* shadowed *with respect to σ if one of the following holds.*

- *$x' > y$, $m < M$, and $\pi_{x'-1} > \pi_M$.*
- *$x' > y$, $m > M$, and $\pi_{x'-1} < \pi_m$.*
- *$y' < x$, $m < M$, and $\pi_{y'+1} < \pi_m$.*

- *$y' < x$, $m > M$, and $\pi_{y'+1} > \pi_M$.*

An entangling *of π is a collection $E = (\sigma_1, \ldots, \sigma_t)$ of pairwise disjoint consecutive subsequences of π, where $\sigma_i = \pi[a_i, b_i]$ for any $1 \leq i \leq t$, satisfying the following.*

- *For any $2 \leq j \leq t$, the following holds. Either $a_j > b_1$ and $\min_{i<j} \min \sigma_i < \pi_{a_j} < \max_{i<j} \max \sigma_i$, or $b_j < a_1$ and $\min_{i<j} \min \sigma_i < \pi_{b_j} < \max_{i<j} \max \sigma_i$.*

- *For any $2 \leq j \leq t$, σ_j is not shadowed with respect to σ_1.*

- *For any $1 \leq \ell \leq k$, there exists $\sigma \in E$ such that $\min \sigma \leq \ell \leq \max \sigma$.*

For the above entangling E of π, define $\Lambda(E)$ as the partition of π in which $\sigma_1, \ldots, \sigma_t$ serve as parts, and any element of π not in $\bigcup_{i=1}^{t} \sigma_i$ has its own part. Denote $d(E) = |\Lambda(E)| = k - \sum_{\sigma \in E}(|\sigma| - 1)$. Finally, the entangling number *of π is $d(\pi) = \max_E\{d(E)\}$ where E ranges over all valid entanglings of π.*

Theorem 5.8. *For any pattern π and entangling E of π, there exists $S \in \{+,-\}^{|E|}$ for which the signed partition $P = (\Lambda(E), S)$ is unique. In particular, $d(\pi) \leq u(\pi)$ for any pattern π.*

The following is an immediate yet important corollary of Theorems 5.5 and 5.8.

Corollary 5.9. *For any pattern π, any non-adaptive one-sided ε-test for π-freeness must make $\Omega\left(\varepsilon^{-1/d(\pi)} n^{1-1/d(\pi)}\right)$ queries.*

A useful simple special case of Corollary 5.9 is the following.

Corollary 5.10. *For a pattern $\pi = (\pi_1, \ldots, \pi_k)$, let $m(\pi) = \max_{1 \leq i \leq k-1} |\pi_{i+1} - \pi_i|$. Then $m(\pi) \leq d(\pi)$, and in particular, any non-adaptive one-sided ε-test for π-freeness must make $\Omega\left(\varepsilon^{-1/m(\pi)} n^{1-1/m(\pi)}\right)$ queries.*

Note that a pattern π of length k with $|\pi^{-1}(1) - \pi^{-1}(k)| = 1$ satisfies $m(\pi) = k-1$, so Theorem 5.2 is actually a special case of Corollary 5.10.

Theorem 5.3 follows from Corollary 5.9 by observing that $d(\pi) \geq k-3$ holds w.h.p. for a random pattern π of length k; actually, both $d(\pi)$ and $u(\pi)$ are concentrated in the values $k-2$ and $k-3$, as $u(\pi) = k-1$ holds with probability $O(1/k)$.

There exist patterns π for which $d(\pi) < u(\pi)$. In particular, partitions with a unique signed form are not necessarily entanglings, so the sufficient condition for uniqueness from Theorem 5.8 is not a necessary one. For example, one can verify that $\pi = (4,1,2,5,6,3)$ satisfies $d(\pi) = 3$ but $u(\pi) = 4$; a unique signed partition of size 4 for π is (Λ, S) where $\Lambda = ((4,1), 2, 5, (6,3))$ and $S = (+,-,-,+))$.

The following necessary condition for being a unique signed partition is easy to prove.

Observation 5.11. *Let π be a pattern of length k and let $P = (\Lambda, S)$ be a unique signed partition for π. Then Λ satisfies the following conditions.*

- *For any $1 \leq \ell \leq k$ there exists $\sigma \in \Lambda$ of length bigger than one, such that $\min \sigma \leq \ell \leq \max \sigma$.*
- *Let $\sigma \in \Lambda$ with $|\sigma| > 1$. If $\max \sigma < k$ then there exists $\sigma' \in \Lambda$ satisfying $\min \sigma' < \max \sigma < \max \sigma'$, and similarly, if $\min \sigma > 1$ then there exists $\sigma' \in \Lambda$ satisfying $\min \sigma' < \min \sigma < \max \sigma'$.*

The size $|\Lambda|$ of the largest partition Λ of π satisfying the conditions in Observation 5.11 might be bigger than the USPN of π. For example, the partition $\Lambda = ((5,1),3,2,7,6,(8,4))$ of the pattern $\pi = (5,1,3,2,7,6,8,4)$ satisfies these conditions, but one can verify that it is not a unique signed partition. By Observation 5.11, none of the other partitions of π of size 6 have a unique signed form, so $u(\pi) < 6 = |\Lambda|$. In fact, $u(\pi) = 5$ in this case, as $((1,3),(2,7),(6,8))$ is an entangling.

Permutation-dependent hierarchy in the non-adaptive case

The statement of Conjecture 5.6 suggests that there is a pattern-dependent hierarchical behavior of the query complexity for one-sided non-adaptive testing of π-freeness as a function of π. The following result verifies that such an hierarchical structure indeed exists.

Theorem 5.12. *For any two positive integers $k \geq 2$ and $1 \leq \ell \leq k-1$, there is a pattern π of length k with $m(\pi) = \ell$, for which the optimal non-adaptive ε-test makes $\tilde{\Theta}_\varepsilon\left(n^{1-1/\ell}\right)$ queries, where the $\tilde{\Theta}_\varepsilon$ notation hides a term polynomial in $\log n$ and ε.*

In particular, we conclude that for any positive integer ℓ, there exist infinitely many patterns π for which the query complexity of one-sided non-adaptive testing of π-freeness is $\tilde{\Theta}_\varepsilon(n^{1-1/\ell})$. This answers and generalizes the open question of Newman et al. [109], who asked whether there exist infinitely many patterns π that have a non-adaptive one-sided test for π-freeness making at most $O(n^{0.99})$ queries (for a fixed ε).

We concluding by commenting that the full version of the work presented here [23] also provides results showcasing a hierarchy of adaptivity for this problem (specifically when the pattern is $\pi = (1,3,2)$), settling an open question by Canonne and Gur [42].

5.1.3 Discussion and Open Problems

The problem of (one-sided) testing for π-freeness demonstrates a wide array of interesting phenomena: An exponential separation between the adaptive and the non-adaptive case, surprising hardness results and pattern-dependent hierarchical behaviors in the non-adaptive case, and a hierarchy of adaptivity that is the first of its kind. We believe that these results serve as a strong motivation to try to achieve a complete understanding of the problem. Below we suggest several possible directions for future research.

The adaptive case Testing π-freeness in the adaptive case is still far from being understood. In particular, the question whether all patterns are testable adaptively with number of queries that is polylogarithmic in n is still wide open, even if we allow for two-sided tests. At this point, this seems to be the most intriguing open question regarding testing π-freeness.

Improving bounds in the non-adaptive case While our understanding of the non-adaptive case is far better than that of the adaptive case, there are still gaps in it. The main goal here is to obtain good pattern-dependent upper bounds: Conjecture 5.6 states that our lower bound is actually tight, and it will be obviously interesting to understand if it holds.

Understanding the USPN Another interesting direction would be to obtain a simple complete combinatorial characterization of the USPN of any given pattern. Currently we have lower and upper bounds for the USPN of a pattern (Theorem 5.8 and Observation 5.11, respectively), that are usually tight for small patterns, and we know that the USPN of a pattern is computable in constant time.

Two-sided testing All known results so far are for one-sided testing, aside from the two-sided lower bound in the partially adaptive setting [23]. It is worth to note that the $\Omega_\varepsilon(n^{1/2})$ lower bound for one-sided testing of all non-monotone patterns can be (carefully) translated into the same bound for two-sided tests. However, the proofs of other one-sided non-adaptive lower bounds do not seem to translate well to the two-sided setting. Therefore, it will be interesting to understand what is the query complexity of optimal *two-sided* tests, both in the adaptive and the non-adaptive case. Specific questions of interest include (but are not limited to) the following: When do the non-adaptive two-sided lower bounds match the one-sided ones? Can one obtain a general two-sided upper bound that beats the tight one-sided upper bound of Theorem 5.1 for patterns of size bigger than three? Does two-sidedness help testing in the adaptive case?

Families of forbidden order patterns It will be interesting to investigate the case where more than one order pattern is forbidden (note that there are families for which the question does not make sense; for example, the famous theorem of Erdős-Szekeres [61] implies that any sequence of length at least $k^2 - 2k + 2$ must contain one of the monotone patterns of length k). As mentioned in [109], all one-sided upper bounds from the single-pattern case carry over to the multiple-pattern case, but the lower bounds do not; for example, there exists a family of two non-monotone patterns of size 3 that has a one-sided non-adaptive test whose query complexity is polylogarithmic in n. Some specific open questions of interest: Is the upper bound from Theorem 5.1 tight in this case? How does the non-adaptive family-dependent hierarchy look like?

Forbidden order patterns in multi-dimensional structures How does π-freeness behave in structures of higher dimensions, such as the hypergrid or the Boolean hypercube? The sample-based upper bound from subsection 5.1.1 still holds in these cases, provided that the input contains many pairwise-disjoint copies of the forbidden structure π. However, in contrast to the one-dimensional case, it is then no longer clear whether being far from π-freeness implies that the input indeed has many pairwise-disjoint π-copies. Interestingly, recent work of Grigorescu, Kumar, and Wimmer [89] gives strong evidence that testing order pattern freeness on the hypercube is hard.

5.2 Upper Bound

In this section we provide the proof of Theorem 5.1. The test that is used to prove the upper bound is one-sided, and indicates that the input sequence $f\colon [n] \to \mathbb{R}$ has a π-copy only if it finds one. Thus, the testing problem reduces to the following search problem: Given query access to un unknown sequence f that is ε-far from π-freeness, the goal is to find a π-copy in f. Here and henceforth, we omit floor and ceiling signs, as they do not make an essential different in the arguments. The proof of Theorem 5.1 follows immediately from the next lemma, which provides a sublinear algorithm to find a π-copy in a sequence f, assuming that f is far enough from π-freeness.

Lemma 5.13. *Let π be a pattern of length $k \geq 3$, and suppose that $f\colon [n] \to \mathbb{R}$ is ε-far from π-freeness for some $\varepsilon \geq c_k n^{-1/9}$, where c_k depends only on k, and n is large enough (as a function of k). Then there is an algorithm that finds, with probability $2/3$, a copy of π in f by querying $O(\varepsilon^{-\frac{1}{k-1}} n^{1-\frac{1}{k-1}})$ entries in f.*

Remark 5.14. *Lemma 5.13 is stronger than what is needed to obtain a one-sided test, in the sense that ε is allowed to scale with n; for the proof of Theorem 5.1 a lemma that applies to a constant ε would have been sufficient. However, the added flexibility of the lemma reflects that the statement of Theorem 5.1 would still be true should we take ε^{-1} as a slowly-growing function of n.*

Proof. The proof idea is as follows. Given an input sequence $f\colon [n] \to \mathbb{R}$, we partition $[n]$ into a collection $\mathcal{I}$ of n/m intervals of size m each, for a suitable choice of m; we may assume, for convenience, that m divides n. Suppose that f is ε-far from π-freeness. Then f contains a set $\mathcal{A}$ of $\varepsilon n/k$ pairwise disjoint π-copies. We consider two cases, where for each of the cases the queries made are different. Our actual algorithm is a combination of the algorithms for each of the cases.

In the first case, most π-copies in $\mathcal{A}$ have at least two entries in the same interval; the algorithm for this case queries a set of whole intervals, chosen uniformly at random, and a set of single elements, also chosen uniformly at random. The analysis of this case does not use the fact that π is a permutation. In the second case, most π-copies in $\mathcal{A}$ do not have two entries in the same interval, and it can be shown that the sampler (which samples entries of f uniformly at random) suffices for this case. Here we do use the fact that π is a permutation, and the analysis actually shows that the required number of queries is much smaller (for constant ε) than in the first case.

We now give the full details. Pick the interval size to be $m = (\varepsilon n)^{1-1/(k-1)}$, and write $\pi = (\pi_1, \dots, \pi_k)$. The π-copies are represented in $\mathcal{A}$ as k-tuples $t = (t_1, \dots, t_k)$ where t_i is the location of the element corresponding to π_i in the copy. Write $\mathcal{A} = \mathcal{B} \cup \mathcal{C}$, where $\mathcal{B}$ contains all π-copies from $\mathcal{A}$ that have at least two entries in the same interval, and $\mathcal{C}$ contains all π-copies that have at most one entry in each interval. Then either $|\mathcal{B}| \geq \varepsilon n/2k$ or $|\mathcal{C}| \geq \varepsilon n/2k$.

Case 1: $|\mathcal{B}| \geq \varepsilon n/2k$ Our algorithm for this case is described as follows. We first pick a set $Q_1 \subseteq \mathcal{I}$ of intervals, where every $I \in \mathcal{I}$ is included in Q_1 with probability $p = cm/\varepsilon n = c(\varepsilon n)^{-1/(k-1)}$, independently of other intervals. Here $c = 100k^2$ depends (polynomially) on k. Next, we pick a set Q_2 of elements from $[n]$, where each element is added to Q_2 with probability p, uniformly and independently of other elements. Up until now, the algorithm does not make any queries.

An independent sampling trick Variants of the following simple idea are used several times along the chapter. Let E_{found} be the event that the subsequence of f induced by Q_1 and Q_2 contains a π-copy. Let E_{big} be the event that $|Q_1| > 100c/\varepsilon$ or $|Q_2| > 100cm/\varepsilon$. By Markov's inequality, $\mathbf{Pr}(E_{\text{big}}) \leq 1/50$. If E_{big} occurs, then the algorithm stops without making any queries (and hence it does not find a π-copy in f). If E_{big} has not occurred, then the algorithm now queries all elements induced by Q_1 and Q_2. Thus, the algorithm finds a π-copy if and only if E_{found} occurs and E_{big} does not occur. The number of queries made is at most $200cm/\varepsilon = O(\varepsilon^{-\frac{1}{k-1}} n^{1-\frac{1}{k-1}})$, as desired. The probability that the algorithm finds a π-copy is at least $\mathbf{Pr}(E_{\text{found}}) - \mathbf{Pr}(E_{\text{big}}) \geq \mathbf{Pr}(E_{\text{found}}) - 1/50$. Thus, it remains to show that $\mathbf{Pr}(E_{\text{found}}) \geq 2/3 + 1/50$ (note that we consider here the unconditional probability of E_{found}, and in particular, we do not condition on E_{big} not happening).

Analyzing the event E_{found} For each $I \in \mathcal{I}$, let t_I denote the number of π-copies from $\mathcal{B}$ that have at least two entries in I, and let $t = \sum_{I \in \mathcal{I}} t_I$, so $\varepsilon n/2k \leq t \leq \varepsilon n$. let X be the random variable that counts the number of π-copies from $\mathcal{B}$ that have at least two entries in some $I \in Q_1$. The expectation of X is $\mathbb{E}[X] = tp \geq cm/2k = 50km$, and the variance of X is bounded by

$$\mathbb{E}[X^2] \leq p \sum_{I \in \mathcal{I}} t_I^2 \leq pm^2 \varepsilon n/m = cm^2$$

where the second inequality follows from convexity arguments, using the facts that $0 \leq t_I \leq m$ for every I and $\sum t_I = t \leq \varepsilon n$. Thus, the standard deviation of X is bounded by $m\sqrt{c} = 10km$. By Chebyshev's inequality, we get that $X \geq m$ with probability at least $9/10$.

Assume now that $X \geq m$, that is, there exists a set $\mathcal{B}' \subseteq \mathcal{B}$ of m π-copies that have at least two of their entries in intervals from Q_1. For each such copy, the event that all other $k-2$ (or less) entries of it are in Q_2 has probability at least $p^{k-2} = c^{k-2}/m$, and is independent of the corresponding events of the other copies in $\mathcal{B}'$. Thus, the probability that none of these events occurs is bounded by $(1 - c^{k-2}/m)^m \leq e^{-c^{k-2}} < 1/100$. This finishes the proof.

Case 2: $|\mathcal{C}| \geq \varepsilon n/2k$ We start with some notation. For a copy $t = (t_1, \ldots, t_k) \in \mathcal{C}$, we define $I_i(t)$ as the interval in $\mathcal{I}$ containing t_i.

Non-extremal π-copies For any interval $I \in \mathcal{I}$, let $y_1 \leq \ldots \leq y_m$ be the elements of $f(I) = \{f(x) : x \in I\}$, and let $y_I^- = y_{\lceil \varepsilon m/6k \rceil}$ and $y_I^+ = y_{\lfloor m - \varepsilon m/6k \rfloor}$. We say that a π-copy $t = (t_1, \ldots, t_k)$ is *top-high* if $f(t_{\pi^{-1}(k)}) > y^+_{I_{\pi^{-1}(k)}(t)}$, and *bottom-low* if $f(t_{\pi^{-1}(1)}) < y^-_{I_{\pi^{-1}(1)}(t)}$. A copy that is neither top-high nor bottom-low is said to be *non-extremal.* In other words, a π-copy is non-extremal if its highest point is not too high with respect to the interval it lies in, and similarly, its lowest point is not too low with respect to its interval. Note that the number of top-high π-copies in $\mathcal{C}$ is bounded by $\varepsilon n/6k$ (as each interval contributes no more than $\varepsilon m/6k$ such copies), and similarly for the number of bottom-low π-copies. Thus, there exists a set $\mathcal{C}' \subseteq \mathcal{C}$ of $\varepsilon n/6k$ non-extremal π-copies.

The main idea is that with sufficiently many queries, the sampler – a sample-based algorithm to find a π-copy – will be able to capture all entries of a non-extremal π-copy $t = (t_1, \ldots, t_k) \in \mathcal{C}'$ besides the lowest entry $t_{\pi^{-1}(1)}$ and the highest entry $t_{\pi^{-1}(k)}$, which will be replaced by a small enough entry from $I_{\pi^{-1}(1)}(t)$ and a large enough entry from $I_{\pi^{-1}(k)}(t)$, respectively. Note that this is a valid π-copy.

Analyzing the sampler Let $p = cm/\varepsilon n = c(\varepsilon n)^{-1/(k-1)}$ as above. Let E_{found} be the event that a subsequence g of f, constructed by putting each entry of f in it with probability p, contains a π-copy. Using the sampling trick from the first case, it is enough to show that $\mathbf{Pr}(E_{\text{found}}) \geq 2/3 + 1/50$. Before we continue, we define the events A_t, B_t, E_t for any π-copy $t = (t_1, \ldots, t_k) \in \mathcal{C}'$ as follows. A_t is the event that all entries $\{t_{\pi^{-1}(j)}\}_{j=2}^{k-1}$ of t are included in g, so $\mathbf{Pr}(A_t) = p^{k-2} = c^{k-2}/m$. B_t is the event that g contains $x \in I_{\pi^{-1}(1)}(t)$ and $x' \in I_{\pi^{-1}(k)}(t)$ such that $f(x) \leq f(t_{\pi^{-1}}(1))$ and $f(x') \geq f(t_{\pi^{-1}}(k))$, so $\mathbf{Pr}(B_t) \geq 1 - 2(1-p)^{\varepsilon m/6k}$. Finally, $E_t = A_t \cap B_t \subseteq E_{\text{found}}$ indicates that g contains a π-copy. Note that any event A_t is independent of all other events, and B_t is only dependent on events $B_{t'}$ where $I_{\pi^{-1}(1)}(t) = I_{\pi^{-1}(1)}(t')$ or $I_{\pi^{-1}(k)}(t) = I_{\pi^{-1}(k)}(t')$; there are at most $2m$ such events for each B_t.

The analysis of patterns of length 3 differs from that of longer ones.

π of length $k > 3$. The probability that none of the events A_t for $t \in \mathcal{C}'$ holds is at most $(1 - p^{k-2})^{|\mathcal{C}'|} \leq \exp\left(-p^{k-2}\varepsilon n/6k\right) = \exp(-c^{k-2}(\varepsilon n)^{1/(k-1)}/6k) < 1/10$. Suppose then that A_t holds for some $t \in \mathcal{C}'$. The probability that B_t does not occur is bounded by $2(1-p)^{\varepsilon m/6k} \leq 2\exp\left(-p\varepsilon m/6k\right) = 2\exp\left(-cm^2/6kn\right) = 2\exp(-\varepsilon^{2-2/(k-1)} n^{1-2/(k-1)} c/6k) < 1/10$ for large enough n. Thus, $\mathbf{Pr}(E_{\text{found}}) = \mathbf{Pr}(\exists t\colon A_t \wedge B_t) > 8/10 > 2/3 + 1/50$ in this case, as desired.

π of length $k = 3$. Let X_t denote the indicator random variable of E_t and let $X = \sum_{t \in \mathcal{C}'} X_t$. For every $t \in \mathcal{C}'$, we have $\mathbf{Pr}(A_t) = p = c/m$ and $\mathbf{Pr}(B_t) \geq (1 - (1-p)^{\varepsilon m/18})^2 \geq (1 -$

$\exp(-p\varepsilon m/18))^2 = (1 - e^{-c\varepsilon/18})^2 \geq c^2\varepsilon^2/400$, where the last inequality holds when $\varepsilon \leq \alpha k^{-2}$ for a small enough α, as $e^{-x} \leq 1 - 9x/10$ for small enough x. Thus, $\mathbb{E}[X] = \sum_{t\in\mathcal{C}'} \mathbf{Pr}(A_t)\,\mathbf{Pr}(B_t) \geq \frac{\varepsilon n}{18}\frac{c}{m}\frac{c^2\varepsilon^2}{400} = \frac{c^3}{7200}\varepsilon^{5/2}n^{1/2}$. On the other hand,

$$\mathbf{Var}(X) = \sum_{t\in\mathcal{C}'}\mathbf{Var}(X_t) + \sum_{t\neq t'\in\mathcal{C}}\mathbf{Cov}(X_t, X_{t'}) \leq \mathbb{E}[X] + 2m|\mathcal{C}'|\max_{t,t'}\mathbf{Pr}(A_t)\,\mathbf{Pr}(A_{t'})\,\mathbf{Pr}(B_t), \quad (5.1)$$

where the inequality builds on the following two facts. The first is that $\mathbf{Cov}(X_t, X_{t'}) \leq \mathbb{E}(X_tX_{t'}) \leq \mathbf{Pr}(A_t)\,\mathbf{Pr}(A_{t'})\,\mathbf{Pr}(B_t)$, as the events $A_t, A_{t'}, B_t$ are mutually independent. The second fact is that, for any $t \in \mathcal{C}'$, $\mathbf{Cov}(X_t, X_{t'}) = 0$ for all but $2m$ of the tuples t', as discussed above.

The second term in (5.1) is bounded by $2c^2\sqrt{\varepsilon n}$. Therefore, the standard deviation of X is bounded by $\sqrt{\mathbb{E}[X]} + \sqrt{2}c\varepsilon^{1/4}n^{1/4} \leq \mathbb{E}[X]/10$, where the bound on ε in the statement of the lemma is chosen so that the last inequality holds (note that $\varepsilon^{1/4}n^{1/4} = \varepsilon^{5/2}n^{1/2}$ for $\varepsilon = n^{-1/9}$, and thus the smallest possible value of ε for which this proof works is $\Theta_k(n^{-1/9})$). Using Chebyshev's inequality, $\mathbf{Pr}(E_{\text{found}}) = \mathbf{Pr}(X > 0) \geq 9/10 > 2/3 + 1/50$, concluding the proof. $\square$

5.3 Lower Bounds

In this section we provide proofs for our lower bounds in the non adaptive case. These are Theorems 5.2, 5.5 and 5.8. We start with the proof of Theorem 5.5. Then, we use it to finish the proof of Theorem 5.2, which requires us to prove a relatively simple special case of Theorem 5.8. Finally, we prove Theorem 5.8 in its full generality. We chose to present the proofs in this order for the sake of readability, as the proof of Theorem 5.8 will be easier to understand after tackling the special case considered in Theorem 5.2.

Proof of Theorem 5.5. Fix a pattern π of length k, and let $P = (\Lambda, S)$ be a unique signed partition of π of size u, where $\Lambda = (\sigma_1, \ldots, \sigma_u)$ and $S = (s_1, \ldots, s_u)$. Let $0 < \varepsilon < \varepsilon_0(k)$ and let $n > n_0(k)$ be an integer, where $\varepsilon_0(k) \leq 1/2k$ is small enough as a function of k and $n_0(k)$ is large enough as a function of k. We may assume, for convenience, that $m = n/k$ is an integer and that εm is an integer bigger than k (translating the result to any n and ε comes at a "price" of a multiplicative constant that depends only on k).

Recall that a one sided ε-test for π-freeness must always accept π-free sequences, and reject sequences that are ε-far from π-freeness with probability at least $2/3$. Thus, any one sided test T for π-freeness must always accept its input if the subsequence it queries is π-free. This follows from the fact that for any π-free sequence $g\colon [q] \to \mathbb{R}$, any $n > q$ and any $1 \leq t_1 < \ldots < t_q \leq n$, there exists a π-free sequence $f\colon [n] \to \mathbb{R}$ such that $f(t_j) = g(j)$ for any $j = 1, \ldots, q$. That is, any π-free queried subsequence might possibly be contained in a π-free sequence, and hence must be accepted by any one-sided test. Therefore, any one-sided test for π-freeness can be seen as a non-adaptive *search algorithm for π in f*, whose goal is to find a π-copy in an unknown input sequence f that is guaranteed to be ε-far from π-freeness, with success probability at least $2/3$.

We use Yao's principle, constructing a family $\mathcal{F}$ of sequences $f\colon [n] \to \mathbb{R}$ that are ε-far from π-freeness, which satisfies the following property for some constant $c_k > 0$. For any $1 \le t_1 < \ldots < t_q \le n$ where $q < c_k \varepsilon^{-1/u} n^{1-1/u}$, it holds that

$$\mathbf{Pr}_{f \in \mathcal{F}} (\text{subsequence of } f \text{ in indices } t_1, \ldots, t_q \text{ contains a } \pi\text{-copy}) < 2/3. \tag{5.2}$$

Combining (5.2) with a standard Yao-type argument completes the proof, as it implies that any (possibly probabilistic) search algorithm for π in f, where f is chosen uniformly at random from $\mathcal{F}$, must make $c_k \varepsilon^{-1/u} n^{1-1/u}$ queries to have success probability $2/3$.

Constructing $\mathcal{F}$ In the rest of the proof, we present a family $\mathcal{F} = \mathcal{F}(P)$ for which (5.2) holds. Let us describe the structure of the sequences $f \in \mathcal{F}$ before diving into the technical details. A sequence $f \in \mathcal{F}$ looks like a blowup of f_P, where each blown up part is planted, starting at a random location, inside a longer interval (making it hard for a non-adaptive test to "guess" where each part is located inside its interval). More specifically, each part σ_i of the partition Λ corresponds to an interval I_i in f whose length is $|\sigma_i| m$. In this interval, there are εn copies of σ_i ordered in an increasing manner if s_i is a $+$, and a decreasing manner if σ_i is a $-$, where each σ_i-copy is a consecutive subsequence of f. The value of f on these σ_i-copies (for each i) is "aligned" with other intervals, so that f contains a set of εn pairwise disjoint π-copies, without containing any other π-copy (here we use the fact that $P = (\Lambda, S)$ is unique). The rest of the elements in each interval are chosen in a manner that does not create any other π-copy. To make $\mathcal{F}$ "random enough," the points where the consecutive copies begin in each interval are chosen uniformly at random. This assures that the probability for each k-tuple of entries in f to induce a π-copy is sufficiently small, proving (5.2).

We now provide the technical details. For every $1 \le i \le u$, write $\sigma_i = \pi[j_{i-1}+1, j_i]$, where $j_0 = 0$ and let $\delta_i = j_i - j_{i-1}$ denote the length of σ_i. For any $1 \le i \le u$, let $I_i = \{m j_{i-1}+1, \ldots, m j_i\}$. A sequence $f\colon [n] \to \mathbb{R}$ is in $\mathcal{F}$ if for any $1 \le i \le u$ there exists $0 \le n_i \le (1 - k\varepsilon)\delta_i m$, such that the following conditions hold.

- For every $1 \le i \le u$ where s_i is a $+$, and every $1 \le l \le \delta_i$ and $0 \le r \le \varepsilon n - 1$, we take $f(m j_{i-1} + n_i + r\delta_i + l) = r + \pi_{j_{i-1}+l}/2k$. We also take $f(x) = -1$ for any $m j_{i-1} + 1 \le x \le m j_{i-1} + n_i$, and $f(x) = n$ for any $m j_{i-1} + n_i + \varepsilon n \delta_i + 1 \le x \le m j_i$.

- For every $1 \le i \le u$ where s_i is a $-$, and every $1 \le l \le \delta_i$ and $0 \le r \le \varepsilon n - 1$, we take $f(m j_{i-1} + n_i + r\delta_i + l) = (\varepsilon n - 1 - r) + \pi_{j_{i-1}+l}/2k$. We also take $f(x) = n$ for any $m j_{i-1} + 1 \le x \le m j_{i-1} + n_i$, and $f(x) = -1$ for any $m j_{i-1} + n_i + \varepsilon n \delta_i + 1 \le x \le m j_i$.

Any $f \in \mathcal{F}$ is ε-far from π-freeness Any such f is ε-far from π-freeness. Indeed, for any $0 \le r \le \varepsilon n - 1$, the subsequence of f consisting of all k entries $x \in [n]$ for which $r < f(x) < r + 1$ is a π-copy, so there is a set $\mathcal{D}_f$ of εn pairwise-disjoint π-copies in f.

Any $f \in \mathcal{F}$ does not contain non-trivial π-copies On the other hand, f does not contain any other (i.e., non-trivial) π-copy. To show this we use the fact that $P = (\Lambda, S)$ is a unique signed partition.

Claim 5.15. *Let $f \in \mathcal{F}$. If f contains a non-trivial π-copy, then it contains a non-trivial copy without the values -1 and m.*

Proof sketch. Recall that $k < \varepsilon m$. The proof follows by applying iteratively the following fact, and its symmetric counterpart. If $t = (t_1, \ldots, t_k)$ is a π-copy in f, and if there exist $0 \leq r < \varepsilon n - 1$ and $i \in [k]$ such that $r - 1 \leq f(t_i) < r$, but there is no $j \in [k]$ for which $r < f(t_j) < r + 1$, then f also contains a π-copy created by the following "lifting process," that replaces all entries with values between $r - 1$ (inclusive) and r (exclusive) with entries whose values are bigger than r and smaller than $r + 1$.

If $r > 0$, we replace any t_i satisfying $r - 1 < f(t_i) < r$ with the unique entry t' satisfying $f(t') = f(t_i) + 1$. If $r = 0$ we replace t_i with the closest entry t' among those satisfying $0 < f(t') < 1$. $\square$

Suppose now to the contrary that f contains a non-trivial π-copy in the entries $x_1 < \ldots < x_k \in n$, without the values -1 and m, and let $R = \{\lfloor f(x_i) \rfloor : 1 \leq i \leq k\} \subseteq \{0, 1, \ldots, \varepsilon n - 1\}$, so $2 \leq |R| \leq k$. We now arbitrarily add elements from $\{0, 1, \ldots, \varepsilon m - 1\}$ to R to obtain a set R' of size *exactly* k.

Let g be the subsequence of f over the set of entries $W(R') = \{w \in [n] : \lfloor f(w) \rfloor \in R'\}$. In particular $x_1, \ldots, x_k \in W(R')$, so g contains a non-trivial π-copy. But this is a contradiction – the nature of our construction (and in particular, the choice of signs) implies that g is order-isomorphic to the sequence f_P given in Definition 5.4, which does not contain non-trivial π-copies (as P is unique). Thus, the only π-copies in f are the trivial copies that come from $\mathcal{D}_f$.

Analysis: $\mathcal{F}$ satisfies desired conditions Finally, we show that the probability for a k-tuple $1 \leq t_1 < \ldots < t_k \leq n$ to induce a π-copy in a sequence $f \in \mathcal{F}$ chosen uniformly at random is sufficiently small. We may restrict ourselves to tuples containing exactly δ_i entries in I_i for any $1 \leq i \leq u$, as these are the only tuples with positive probability to induce a π-copy. Suppose that $f \in \mathcal{F}$ contains a π-copy in entries $t_1 < \ldots < t_k$. This copy must come from $\mathcal{D}_f$, and so there exists some $0 \leq r \leq \varepsilon n - 1$ such that $r < f(t_l) < r + 1$ for any $1 \leq l \leq k$. The values of r and $t_{j_1}, t_{j_2}, \ldots, t_{j_u}$ determine $n_1, \ldots, n_u$ uniquely. In other words, f is the only sequence, among all $|\mathcal{F}| > (n/2k)^u \geq (2k)^{-k} n^u$ sequences from $\mathcal{F}$, that has a π-copy of height between r and $r + 1$ whose j_i-th entry lies in t_{j_i}, for any $1 \leq i \leq u$. In total, only at most εn such possible choices $f \in \mathcal{F}$ have a π-copy whose j_i-th entry lies in t_{j_i}, for any $1 \leq i \leq u$. Thus, we have:

$$\mathbf{Pr}_{f \in \mathcal{F}}(\text{subsequence of } f \text{ in indices } t_{j_1}, \ldots, t_{j_u} \text{ is contained in a } \pi\text{-copy}) \leq \frac{\varepsilon n}{|\mathcal{F}|} < \frac{(2k)^k \varepsilon}{n^{u-1}} \quad (5.3)$$

We are now ready to finish the proof of (5.2). Pick $c_k = (3k)^{-k/u}$, and let $t = (t_1, \ldots, t_q)$ with $1 \leq t_1 < \ldots < t_q \leq n$ be a q-tuple, where $q < c_k \varepsilon^{-1/u} n^{1-1/u}$. t contains $\binom{q}{u} \leq q^u$ u-tuples, so by a union bound, the expected number of u-tuples contained in a π-copy (over a uniform choice $f \in \mathcal{F}$) is less than $(2k)^k \varepsilon q^u n^{1-u} < 2/3$. Thus, the probability that the subsequence of f on t contains a π-copy is less than $2/3$, as desired. □

Proof of Theorem 5.2. Using Theorem 5.5, it is enough to show that $u(\pi) = k - 1$ for any pattern π of length k satisfying $|\pi^{-1}(1) - \pi^{-1}(k)| = 1$. Let $\pi = (\pi_1, \ldots, \pi_k)$ be a pattern of length k, and assume, without loss of generality, that $\pi_\ell = 1$ and $\pi_{\ell+1} = k$ for some $1 \leq \ell \leq k-1$. We take the following signed partition $P = (\Lambda, S)$ of size $k-1$. $\Lambda = (\sigma_1, \ldots, \sigma_{k-1})$ where σ_i consists of the single element π_i for any $i < \ell$, $\sigma_\ell = (1, k)$, and σ_i is the single element π_{i+1} for any $i > \ell$. The sign vector $S = (s_1, \ldots, s_{k-1})$ is defined as follows. s_ℓ is a $-$, and for any $i \neq \ell$, s_i is a $+$ if and only if $\pi_i > \pi_{i+1}$.

We now show that P is unique, implying that $u(\pi) \geq |P| = k - 1$, as needed. Consider the sequence $f = f_P$, as defined in Definition 5.4. We partition the entries of f_P into intervals $I_1, \ldots, I_{k-1}$, where I_i contains all entries that participate in the σ_i-part of some π-copy in f_P. In other words, $I_i = \{(i-1)k+1, \ldots, ik\}$ for any $1 \leq i < \ell$, $I_\ell = \{(\ell-1)k+1, \ldots, (\ell+1)k\}$ and $I_i = \{ik+1, \ldots, (i+1)k\}$ for any $\ell < i \leq k-1$.

Let $q = (q_1, \ldots, q_\ell)$ be a π-copy in f_P. The following claim sheds light on the structure of q with respect to the intervals $I_1, \ldots, I_{k-1}$.

Claim 5.16. *For any $i = 1, \ldots, k$ let $\operatorname{ind}(i)$ denote the index of the interval containing q_i, that is, $q_i \in I_{\operatorname{ind}(i)}$. Then $\operatorname{ind}(i) \geq i$ for any $i \leq \ell$ and $\operatorname{ind}(i) \leq i - 1$ for any $i \geq \ell + 1$.*

Proof. Suppose to the contrary that $\operatorname{ind}(i) < i$ for some $i \leq \ell$, and consider the smallest i satisfying this. Then $\operatorname{ind}(i-1) = \operatorname{ind}(i) = i-1$, that is, $q_{i-1}, q_i \in I_{i-1}$. This is a contradiction: If $\pi_i > \pi_{i-1}$ then s_{i-1} is a $-$, so the subsequence of f restricted to I_{i-1} is decreasing, contradicting the fact that q is a π-copy, that must satisfy $f(q_i) > f(q_{i-1})$ since $\pi_i > \pi_{i-1}$. If $\pi_i < \pi_{i-1}$ then s_{i-1} is a $+$ and, symmetrically, we have a contradiction. Thus, $\operatorname{ind}(i) \geq i$ for any $i \leq \ell$. The proof that $\operatorname{ind}(i) \leq i - 1$ for any $i \geq \ell + 1$ is symmetric. □

As a special case of Claim 5.16, we conclude that $q_\ell, q_{\ell+1} \in I_\ell$ for any π-copy $q = (q_1, \ldots, q_k)$. This implies that $f_P(q_\ell) = r + 1/2k$ and $f_P(q_{\ell+1}) = r + 1/2$ for some integer r (since the only length-2 subsequences of f inside I_ℓ that are increasing are $(r + 1/2k, r + 1/2)$, for any integer $0 \leq r \leq k-1$). Hence, for any $i \neq \ell, \ell+1$, q_i must be the unique entry satisfying $f(q_i) = r + \pi_i/2k$. We conclude that f_P does not contain non-trivial π-copies, so P is unique. □

The proof of Theorem 5.8 is based on ideas that are similar to those of the proof of Theorem 5.2, and in particular, a generalized form of Claim 5.16 serves as an important tool in the proof.

Proof of Theorem 5.8. Let π be a pattern of length k, and let $E = \{\tau_1, \ldots, \tau_t\}$ be an entangling of π whose resulting partition $\Lambda = \Lambda(E) = (\sigma_1, \ldots, \sigma_d)$ is of size $d = d(\pi)$. For any $1 \leq \ell \leq t$, denote by $\lambda(\ell)$ the unique index satisfying $\tau_\ell = \sigma_{\lambda(\ell)}$. For any $1 \leq i \leq d$, write $\sigma_i = \pi[j_{i-1}+1, j_i]$ where $j_0 = 0$ and $j_d = k$. We choose the sign vector $S = (s_1, \ldots, s_d)$ as follows.

- For any $1 \leq i \leq d$ where σ_i contains more than one element, s_i is a $+$ if $\min \sigma_i$ lies after $\max \sigma_i$ in π, and otherwise s_i is a $-$.
- For any $i < \lambda(1)$ where σ_i is a single element, s_i is a $+$ if and only if $\pi_{j_i} > \pi_{j_i+1}$.
- For any $i > \lambda(1)$ where σ_i is a single element, s_i is a $+$ if and only if $\pi_{j_i} < \pi_{j_i-1}$.

To finish the proof, we shall show that the signed partition $P = (\Lambda, S)$ is unique, implying that $u(\pi) \geq d = d(\pi)$. For this, we need to show that $f = f_P$ does not contain non-trivial π-copies. As in the proof of Theorem 5.2, for any $1 \leq i \leq d$ let $I_i = \{kj_{i-1}+1, \ldots, kj_i\}$. Let $q = (q_1, \ldots, q_k)$ be a π-copy in f_P. The following claim is the equivalent of Claim 5.16 for our more general case.

Claim 5.17. *For any $i = 1, \ldots, k$ let $\operatorname{ind}(i)$ denote the index for which $q_i \in I_{\operatorname{ind}(i)}$. Then $\operatorname{ind}(j_{i-1}+1) \geq i$ for any $i \leq \lambda(1)$ and $\operatorname{ind}(j_i) \leq i$ for any $i \geq \lambda(1)$.*

Proof. We shall prove the claim for $i \leq \lambda(1)$, as the proof for $i \geq \lambda(1)$ is symmetric. Suppose to the contrary that there exists $i \leq \lambda(1)$ for which $\operatorname{ind}(j_{i-1}+1) < i$, and consider the smallest such i. Then $\operatorname{ind}(j_{i-2}+1) \geq i-1$, and so $\operatorname{ind}(j) = i-1$ for any $j_{i-2}+1 \leq j \leq j_{i-1}+1$.

We show that I_{i-1} does not contain a copy of $\pi[j_{i-2}+1, j_{i-1}+1]$, leading to a contradiction. If $|\sigma_{i-1}| = 1$ then the choice of the sign s_{i-1} is a $+$ if $\pi_{j_{i-1}+1} < \pi_{j_{i-1}}$, and a $-$ otherwise; in the first case, the entries in I_{i-1} are increasing and so it cannot contain $\pi[j_{i-i}, j_{i-1}+1]$, which is a decreasing sequence, a contradiction. In the other case we also get a contradiction, symmetrically. Thus, from here onward we may assume that σ_{i-1} contains more than one element.

The choice of the sign s_{i-1} implies that the only σ_{i-1}-copies in the subsequence of f_P on the interval I_{i-1} are the trivial ones, i.e., those that contain all $|\sigma_{i-1}|$ elements between r and $r+1$ for some integer $0 \leq r \leq k-1$. Thus, we may assume that $r < f_P(q_j) < r+1$ for any $j_{i-2}+1 \leq j \leq j_{i-1}$.

Without loss of generality, assume that s_{i-1} is a $+$; this corresponds to the case where $\max \sigma_{i-1}$ lies before $\min \sigma_{i-1}$ in π. Since E is an entangling, we know that σ_{i-1} is not shadowed with respect to $\sigma_{\lambda(1)}$. This means that $\pi_{j_{i-1}+1} < \max \sigma_{i-1}$, and so $f_P(q_{j_{i-1}+1}) < r+1$. But this contradicts the fact that $\operatorname{ind}(j_{i-1}+1) = i-1$: All $|\sigma_{i-1}|$ entries in I_{i-1} whose value is between r and $r+1$ are assigned to $q_{j_{i-2}+1}, \ldots, q_{j_{i-1}}$, and all entries x of I_{i-1} that come after these entries satisfy $f_P(x) > r+1$. In particular, $q_{j_{i-1}+1} \in I_{i-1}$ so $f_P(q_{j_{i-1}+1}) > r+1$, a contradiction. ☐

To show that q is a trivial π-copy, we prove the following claim by induction.

Claim 5.18. *There exists an integer $r = r(q)$ where $0 \leq r \leq k-1$, satisfying the following. For any $1 \leq l \leq t$, and any $j_{\lambda(\ell)-1}+1 \leq j \leq j_{\lambda(\ell)}$, it holds that $q_j \in I_{\lambda(\ell)}$, and more specifically, $f_P(q_j) = r + \pi_j/2k$.*

Proof. The proof is by induction on ℓ. By Claim 5.17, $q_{j_{\lambda(1)-1}+1}, \dots, q_{j_{\lambda(1)}} \in I_{\lambda(1)}$. By our choice of the sign $s_{\lambda(1)}$, there must be some integer $0 \leq r \leq k-1$, such that for any $j_{\lambda(1)-1} + 1 \leq j \leq j_{\lambda(1)}$, q_j is the unique entry of f_P satisfying $f_P(q_j) = r + \pi_j/2k$. This settles the case $\ell = 1$.

Now let $\ell > 1$, and assume that $f_P(q_j) = r + \pi_j/2k$ for any $j_{\lambda(\ell')-1} + 1 \leq j \leq j_{\lambda(\ell')}$ where $1 \leq \ell' < \ell$. We need to show that $f_P(q_j) = r + \pi_j/2k$ for any $j_{\lambda(\ell)-1} + 1 \leq j \leq j_{\lambda(\ell)}$.

For any $j', j'' \in [k]$ for which we already know that $f_P(q_{j'}) = r + \pi_{j'}/2k$, $f_P(q_{j''}) = r + \pi_{j''}/2k$, and $\pi_{j'} < \pi_{j''}$, it must be true that $f_P(q_j) = r + \pi_j/2k$ for any j satisfying $\pi_{j'} < \pi_j < \pi_{j''}$. To see this, note that the number of entries of f_P with value between $f_P(q_{j'})$ and $f_P(q_{j''})$ (not including $f_P(q_{j'}), f_P(q_{j''})$ themselves) is exactly $\pi_{j''} - \pi_{j'} - 1$. Since q is a π-copy, it also contains exactly $\pi_{j''} - \pi_{j'} - 1$ entries with value between $f_P(q_{j'})$ and $f_P(q_{j''})$, so these entries of q must be precisely all entries of f_P whose value lies in this range.

Without loss of generality, assume that $\lambda(\ell) < \lambda(1)$ (that is, $\tau_l = \sigma_{\lambda(\ell)}$ lies before $\tau_1 = \sigma_{\lambda(1)}$ in π). Since E is an entangling, we know that $\pi_{j'} < \pi_{j_{\lambda(\ell)}} < \pi_{j''}$ for some $\pi_{j'}, \pi_{j''} \in \bigcup_{\ell' < \ell} \tau_{\ell'}$. By the previous paragraph, $f_P(q_{j_{\lambda(\ell)}}) = r + \pi_{j_{\lambda(\ell)}}/2k$, also implying that $\text{ind}(j_{\lambda(\ell)}) = \lambda(\ell)$. By Claim 5.17, $\text{ind}(j_{\lambda(\ell)-1}+1) \geq \lambda(\ell)$, so we get that $\text{ind}(j) = \lambda(\ell)$ for any $j_{\lambda(\ell)-1}+1 \leq j \leq j_{\lambda(\ell)}$. Considering our choice of the sign s_l, it follows that $f_P(q_j) = r + \pi_j/2k$ must hold for any $j_{\lambda(\ell)-1} + 1 \leq j \leq j_{\lambda(\ell)}$. This concludes the inductive proof. □

With Claim 5.18 it is easy to finish the proof. Since E is an entangling, there exist $1 \leq \ell, \ell' \leq d$ such that $1 \in \tau_l = \sigma_{\lambda(\ell)}$ and $k \in \tau_{\ell'} = \sigma_{\lambda(\ell')}$, implying that $f_P(q_{\pi^{-1}(1)}) = r + 1/2k$ and $f_P(q_{\pi^{-1}(k)}) = r + 1/2$ for some $0 \leq r \leq r+1$. Thus, $r < f_P(q_j) < r+1$ for any $1 \leq j \leq k$. Since there are exactly k entries $x \in [k^2]$ for which $r < f_P(x) < r+1$, q must be a trivial π-copy. Therefore, P is unique. □

We finish with an (easy) proof of Theorem 5.3 that builds on Corollary 5.9.

Proof of Theorem 5.3. Let $\pi = (\pi_1, \dots, \pi_k)$ be a pattern of length k chosen uniformly at random. Without loss of generality assume that $\pi_i = 1, \pi_j = k$ for some $i < j$. The probability that $\pi_{i+1} \leq k^{3/4}$ or $\pi_{j-1} \geq k - k^{3/4}$ or $|i-j| < k^{3/4}$ is $O(k^{-1/4})$. Conditioning on the event that none of the above happens, the probability that there exists no $i+1 < x < j-1$ for which $\pi_x < k^{3/4}$ and $\pi_{x+1} > k - k^{3/4}$ is also bounded by $O(k^{-1/4})$ (it is actually exponentially smaller than that). If none of these events happens, then $d(\pi) \geq k-3$, as there exists some $i < x < y$ for which $((\pi_x, \pi_{x+1}), (1, \pi_{i+1}), (\pi_{j-1}, k))$ is an entangling. Indeed, $(1, \pi_{i+1})$ and (π_{j-1}, k) cannot be shadowed with respect to (π_x, π_{x+1}), and the two other conditions of an entanglement hold since $\pi_x < \pi_{i+1}, \pi_{j-1} < \pi_{x+1}$. Thus $d(\pi) \geq k-3$ with probability at least $1 - O(k^{-1/4})$, as desired.

As an added bonus, note that $\mathbf{Pr}(d(\pi) \geq k-2) \geq 19/24 - O(1/k)$: Suppose that $i > 1$, $j < n$, and $j \geq i+2$ (all of these hold with probability $1 - O(1/k)$). Consider the event where either $\max\{\pi_{i-1}, \pi_{i+1}\} \geq \pi_{j-1}$ or $\min\{\pi_{j-1}, \pi_{j+1}\} \leq \pi_{i+1}$. This event has probability 19/24, and if it occurs, one can verify that $d(\pi) \geq k-2$. □

Part III

Understanding Locality in Structured Property Testing

Chapter 6

Testing Local Properties: Follow the Boundaries

The results in this chapter appear in [21].

6.1 Introduction

In this chapter we focus on testing of local properties in structured data. The objects we consider are *d-dimensional arrays*, where d is a positive integer, viewed as a constant. A d-dimensional *array* of *width* n, or an $[n]^d$*-array* in short, is a function $A\colon [n]^d \to \Sigma$ from the hypergrid $[n]^d$ to the *alphabet* Σ, where the alphabet Σ is allowed to be any (arbitrarily large) finite set; we stress that the size of Σ is usually *not required* to be bounded as a function of the other parameters. For example, a *string* is an $[n]^1$-array, and the commonly used RGB representation of images is basically an $[n]^2$-array over $\{0, 1, \ldots, 255\}^3$, where the three values corresponding to each pixel represent the intensity of red, green and blue in it.

We call a property *local* if it can be characterized by a family of small forbidden consecutive patterns. Here, a $[k]^d$-array S is a (consecutive) *subarray* of an $[n]^d$-array A in *location* $(i_1, \ldots, i_d) \in [n-k+1]^d$ if $A(i_1 + j_1 - 1, \ldots, i_d + j_d - 1) = S(j_1, \ldots, j_d)$ for any $j_1, \ldots, j_d \in [k]$. Formally, a property $\mathcal{P}$ of $[n]^d$-arrays over an alphabet Σ is k*-local* (for $2 \le k \le n$) if there exists a family $\mathcal{F}$ of $[k]^d$-arrays over Σ so that the following holds for any $[n]^d$-array A over Σ:

$$A \text{ satisfies } \mathcal{P} \iff \text{None of the (consecutive) subarrays of } A \text{ is in } \mathcal{F}.$$

For $\mathcal{P}$ as above, we sometimes write $\mathcal{P} = \mathcal{P}(\mathcal{F})$ to denote that $\mathcal{P}$ is defined by the forbidden family $\mathcal{F}$. As we shall see soon, many interesting properties of arrays (including a large fraction of the array properties that were previously investigated in the literature) can be characterized this way.

The main contribution of this chapter is a generic one-sided error non-adaptive framework to test k-local properties. In some cases, our method either matches or beats the best known upper bounds on the query complexity (although the running time might be far from optimal in general). We show the optimality of our method by proving a matching lower bound for non-adaptive one-sided tests, as well as a (weaker) lower bound for two-sided tests.

In order to demonstrate the wide range of properties captured by the above definition, we now present various examples of properties that are k-local for small k, including some of the most widely investigated properties in the property testing literature, as well as properties from areas of computer science that were not systematically studied in the context of property testing. In what follows, the sum of two tuples $x = (x_1, \ldots, x_d), y = (y_1, \ldots, y_d)$ is defined as the tuple $(x_1 + y_1, \ldots, x_d + y_d)$; additionally, e^i denotes the i-th unit vector in d dimensions.

Monotonicity Perhaps the most thoroughly investigated property in the testing literature: see e.g. the entries related to monotonicity testing in the Encyclopedia of Algorithms [43, 116] and the references within. An $[n]^d$-array A over an ordered alphabet Σ is *monotone* (non-decreasing) if $A(x) \leq A(y)$ for any $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ satisfying $x_i \leq y_i$ for any i. Monotonicity is 2-local: an array A is monotone if and only if there is no pair $x, x + e^i \in [n]^d$ so that $A(x) > A(x + e^i)$.

Lipschitz continuity Another well-investigated property with connections to differential privacy [16, 32, 44, 92], an $[n]^d$-array A is *c-Lipschitz continuous* if $|A(x) - A(y)| \leq c \sum_{i=1}^{d} |y_i - x_i|$ for any $x, y \in [n]^d$. This condition holds iff $|A(x) - A(x + e^i)| \leq c$ for any $x, x + e^i \in [n]^d$, and thus Lipschitz continuity is also 2-local.

Convexity Discrete convexity is an important geometric property with connections to optimization and other areas [30, 31, 36, 51, 112, 114]. A one-dimensional array A is *convex* if $\lambda A(x) + (1 - \lambda)A(y) \geq A(\lambda x + (1 - \lambda)y)$ for any $x, y \in [n]$ and $0 < \lambda < 1$ satisfying $\lambda x + (1 - \lambda)y \in [n]$. Convexity is 3-local for the case $d = 1$: an array $A : [n] \to \Sigma$ is convex if and only $A[x] - 2A[x + 1] + A[x + 2] \geq 0$ for any $x \in [n - 2]$. In higher dimensions, several different notions of discrete convexity have been used in the literature – see e.g. the introductory sections of the book of Murota on discrete convex analysis [105]. Two of the commonly used definitions, $M^\natural$-convexity and $L^\natural$-convexity, are 3-local and 4-local, respectively: see Theorems 4.1 and 4.2 in [103], where it is shown that both notions can be defined locally using slight variants of the Hessian matrix consisting of the partial discrete derivatives. Another common definition that is a natural variant of the continuous case states that convexity is equivalent to the positive semi-definiteness of the Hessian matrix; under this definition, convexity is 3-local. A strictly weaker notion of convexity, called *separate convexity* [36], is defined as follows: an $[n]^d$-array A is separately convex if it is convex along each of the axes. Similarly to one-dimensional convexity, separate convexity is 3-local for any d.

Properties of higher order derivatives More generally, *any* property of arrays that can be characterized by "forbidden pointwise behavior" of the first k discrete derivatives [36] is

$(k+1)$-local. Monotonicity (for $k=1$), Lipschitz continuity ($k=1$) and convexity ($k=2$) are special cases of such properties.

Submodularity Another important property closely related to convexity [32, 34, 112, 127]. Given $x=(x_1,\ldots,x_d), y=(y_1,\ldots,y_d) \in [n]^d$, define $x \wedge y = (\min(x_1,y_1),\ldots,\min(x_d,y_d))$ and $x \vee y = (\max(x_1,y_1),\ldots,\max(x_d,y_d))$. An $[n]^d$-array is *submodular* if $A[x \wedge y] + A[x \vee y] \leq A[x] + A[y]$ for any $x, y \in [n]^d$. Submodularity is 2-local: one can verify that submodularity is equivalent to the condition that $A(x) + A(x+e^i+e^j) \leq A(x+e^i) + A(x+e^j)$ for all x.

Pattern matching and computer vision Tasks involving pattern matching under some limitations – such as noise in the image, obstructed view, or rotation of elements in the image – are at the core of computer vision and its applications. For example, the local property of not containing a good enough ℓ_1-approximation of a given forbidden pattern is of practical importance in computer vision. Sublinear approaches closely related to property testing are known to be effective for problems of this type, see e.g. [99].

Computational biology Many problems in computational biology are closely related to one-dimensional pattern matching. As an example, a defensive mechanism of the human body against RNA-based viruses involves "cutting" a suspicious RNA fragment, if it finds one of a (small) family of short forbidden consecutive patterns in it, indicating that this RNA might belong to a virus. Thus, in order to generate fragments of RNA that are not destroyed by such defensive mechanisms (which is a basic task in computational biology), understanding the process of "repairing" a fragment so that it will not contain any of the forbidden patterns is an interesting problem related to property testing.

6.1.1 Previous Results on Local Properties

One-dimensional arrays A seminal result of Ergün et al. [63] shows that for constant ε, monotonicity is ε-testable over the line (that is, for one-dimensional arrays) using $O(\log n)$ queries over general alphabets. The non-adaptive one-sided error test proposed in [63] is based, roughly speaking, on imitating a binary search non-adaptively. It was shown by Fischer [66] that the above is tight even for two-sided error adaptive tests, proving a matching $\Omega(\log n)$ lower bound. Later on, Parnas, Ron and Rubinfeld [112] and Jha and Raskhodnikova [92] showed that the $O(\log n)$ upper bound on the non-adaptive one-sided query complexity also holds for convexity and Lipschitz continuity, respectively. For general ε, the upper bound in [92] is of the type $O(\varepsilon^{-1}\log n)$; the same work also presents a matching lower bound of $\Omega(\log n)$ for the one-sided non-adaptive case, while $\Omega(\log n)$ lower bounds for two-sided non-adaptive tests of convexity, and more generally, monotonicity of the ℓ-th derivative, are proved by Blais, Raskhodnikova and Yaroslavtsev [36] using a communication complexity based approach [35]. Finally, a recent result of Belovs [18] refines the one-sided non-adaptive query complexity of monotonicity to $O(\varepsilon^{-1}\log \varepsilon n)$.

When the alphabet is binary (of size two), general positive results are known regarding the testability of local properties in one dimension. It follows from the testability of regular languages, established by Alon et al. [10], that any k-local property is testable in $O(c(\mathcal{F})\varepsilon^{-1}(\log^3(\varepsilon^{-1})))$ queries, where $c(\mathcal{F})$ depends only on the family $\mathcal{F}$ of forbidden consecutive length-k patterns defining the property. However, $c(\mathcal{F})$ can be exponential in k in general.

Multi-dimensional arrays Chakrabarty and Seshadhri [45] extended some of the above results to hypergrids, showing that a general class of so-called "bounded derivative" properties (all of which are 2-local), including monotonicity and Lipschitz continuity as special cases, are all testable over $[n]^d$-arrays with $O(\varepsilon^{-1}d\log n)$ queries. Another work by the same authors [46] shows a matching lower bound of $\Omega(\varepsilon^{-1}d\log\varepsilon n)$ for monotonicity, that holds even for two-sided adaptive tests, while the communication complexity approach of [36] gives a (non-adaptive, two-sided) $\Omega(d\log n)$ lower bound for convexity, separate convexity and Lipschitz.

Submodularity is testable for $d = 2$ with $O(\log^2 n)$ queries [112]; However, no non-trivial upper bound on the query complexity is known for submodularity in the case $d > 2$ and convexity in the case $d > 1$ under the Hamming distance and over general alphabets (although [30] proves constant-query testability for 2D convexity over a binary alphabet). Under L_1-distance and for any d, it was shown in [32] that convexity in $[n]^d$-arrays is testable with number of queries depending only on d.

6.1.2 Our Results

In this chapter, we present a generic approach to test *all* k-local properties of $[n]^d$-arrays. Among other consequences, a simple special case of our result in the one-dimensional regime shows that the abundance of properties whose query complexity is $\Theta(\log n)$ is not a coincidence: in fact, *any* $O(1)$-local property of one-dimensional arrays is testable with $O(\log n)$ queries, using a canonical binary search like querying scheme. In the full version of the results presented here [21], we prove matching lower bounds in $d > 1$ dimensions.

Our first main result is an upper bound on the number of queries required to test any k-local property of $[n]^d$-arrays non-adaptively with one-sided error. The test is *canonical* in a strong sense: The queries it makes depend on n, d, k, and (relatively weakly) on ε; they do not depend on $\mathcal{P}$ or the alphabet Σ. In other words, it makes the same type of queries for all k-local properties of $[n]^d$-arrays over any finite (and not necessarily bounded-size) alphabet.

Theorem 6.1. *Let $2 \le k \le n$ and $d \ge 1$ be integers, and let $\varepsilon > 0$. Any k-local property $\mathcal{P}$ of $[n]^d$-arrays over any finite (and not necessarily bounded size) alphabet has a one-sided error non-adaptive ε-test whose number of queries is*

- $O(\frac{k}{\varepsilon} \cdot \log\frac{\varepsilon n}{k})$ *for $d = 1$.*
- $O(c^d \frac{k}{\varepsilon^{1/d}} \cdot n^{d-1})$ *for $d > 1$.*

Here, $c > 0$ is an absolute constant. The test chooses which queries to make based only on the values of n, d, k, ε, and independently of the property $\mathcal{P}$ and the alphabet Σ.

Note that we are interested here in the domain where n is large and d is considered a constant. Thus, we did not try to optimize the c^d term in the second bullet, seeing that it is negligible compared to n^{d-1} anyway.

Running time The main drawback of our approach is the running time of the test, which is high in general. After making all of its queries, our test runs an *inference* step, where it tries to evaluate (by enumerating over all relevant possibilities) whether a violation of the property must occur in view of the queries made, and reject if this is the case.

Without applying any property-specific considerations, the running time of the inference step is of order $|\Sigma|^{O(n^d)}$. However, for various specific properties of interest, such as monotonicity and 1D-convexity, it is not hard to make the running time of the inference step of the same order of magnitude as the query complexity. Moreover, in one dimension we can use dynamic programming to achieve running time that is significantly better than the naive one, but still much higher than the query complexity in general: $O(|\Sigma|^{O(k)}n)$. This works for any k-local property in one dimension; see the last part of Section 6.1.3 for more details.

Proximity oblivious test Interestingly, the behavior of the test depends quite minimally on ε, and it can be modified very slightly to create a proximity oblivious test (POT) for any k-local property. The useful notion of a POT, originally defined by Goldreich and Ron [85], refers to a test that does *not* receive ε as an input, and whose success probability for an input not satisfying the property is a function of the Hamming distance of the input from the property.

Theorem 6.2. *Fix $d > 0$. Any k-local property $\mathcal{P}$ of $[n]^d$-arrays over any finite (but not necessarily bounded size) alphabet has a one-sided error non-adaptive proximity oblivious test whose number of queries is $O(k \log(n/k))$ if $d = 1$ and $O(kn^{d-1})$ if $d > 1$. For any input A not satisfying $\mathcal{P}$, the rejection probability of A is linear (for fixed d) in the Hamming distance of A from $\mathcal{P}$.*

One can run $O(c_d/\varepsilon)$ iterations of the POT to obtain a standard one-sided non-adaptive test. The query complexity is $O(k\varepsilon^{-1}\log(n/k))$ for $d = 1$ and $O(c_d k\varepsilon^{-1}n^{d-1})$ for $d > 1$, where $c_d > 0$ depends only on d. Thus, the POT-based test is sometimes as good as the test of Theorem 6.1 (specifically, for $d = 1$ it matches the above bounds for almost the whole range of ε and k). In any case, the multiplicative overhead of the POT-based test is sublinear in $1/\varepsilon$ across the whole range.

Type of queries In one dimension, many of the previously discussed properties, including, for example, monotonicity and Lipschitz continuity, are testable in $O(\log n)$ queries (see Section 6.1.1 for a more extensive discussion). Previously known tests for monotonicity and Lipschitz continuity make queries that resemble a binary search in some sense: these tests query pairs of entries of distance 2^i for multiple choices of $0 \leq i \leq \log n$. Our test continues the line of works using querying schemes roughly inspired by binary search. The test queries structures that can be viewed, intuitively, as L_∞-*spheres* of different sizes in $[n]^d$. For this purpose, an L_∞-sphere with radius r and width ℓ in $[n]^d$ is a set $X_1 \times X_2 \times \ldots \times X_d \subseteq [n]^d$, where each X_i is a union of intervals of the form $[a_i, a_i+1, \ldots, a_i+\ell-2, a_i+\ell-1] \cup [b_i-\ell+2, b_i-\ell+3, \ldots, b_i-1, b_i]$, and $b_i - a_i \in \{r, r+1\}$ for any $i \in [d]$. More specifically, our test for k-local properties queries spheres with width $k-1$ and radius of order 2^i for different values of i. In the simple special case where $d = 1$ and $k = 2$, this is very similar to the querying scheme mentioned in the previous paragraph.

Implications In one dimension, the query complexity of the test matches the best known upper bounds (and, in some regimes, refines the dependence on ε) for several previously investigated properties including monotonicity, Lipschitz continuity and convexity. For monotonicity of k-th order derivatives, which is $(k+1)$-local, it proves the first sublinear upper bound on the query complexity: $O(k \log n)$; in comparison, the best known lower bound [36] is $\Omega(\log n)$.

For pattern matching type properties in 1D arrays (including applications in computational biology and other areas), our approach gives a property- and alphabet-independent upper bound of $O(k \log n)$ on the query complexity, with essentially optimal dependence on ε as well. Previously known approaches for testing such properties, like the regular languages testing approach [10], yield tests whose query complexity is dependent on the family of forbidden patterns considered, whose size might be exponential in the locality parameter k. Our approach, on the other hand, requires an $O(\log n)$ "overhead", but its query complexity is independent of the size of the forbidden family discussed. Instead, the dependence in k is linear.

In multiple dimensions, our approach is far from tight for well-understood properties such as monotonicity and Lipschitz continuity, whose query complexity is known to be $\Theta(d \log n)$ (in comparison, our approach yields an $O(n^{d-1})$ type bound). However, for testing of other properties like convexity (for $d > 1$) and submodularity (for $d > 2$) in $[n]^d$-arrays, no non-trivial upper bounds on the query complexity are known over general alphabets, so our upper bound of $O(n^{d-1})$ is the first such bound. While we do not believe this bound is tight in general, this might be a first step towards the development of new tools for efficiently testing such properties.

Sketching for testing The fact that the queries made are completely independent of the property suggests the following sketching technique allowing for "testing in retrospect": Given ε and k in advance, we make all queries of the generic ε-test for k-local properties in "real time", and store them for postprocessing. This is suitable, for example, in cases where we have limited access to a large input for a limited amount of time (e.g. when reading the input requires specialized expensive machinery), but the postprocessing time is not an issue. Note that for this approach we do not need to know the property of interest in advance.

6.1.3 Proof Ideas and Techniques

Here we present the main ideas of our proofs in an informal way. For simplicity, we stick to the one-dimensional case, and assume that ε is fixed and $k = o(n)$.

Upper bound for 1D Suppose that $\mathcal{P} = \mathcal{P}(\mathcal{F})$ is a k-local property of $[n]^1$-arrays A over an alphabet Σ, defined by the forbidden family $\mathcal{F}$. Let S be a consecutive subarray of A of length at least $2k-2$. The *boundary* of S consists of the first $k-1$ elements and the last $k-1$ elements of S, and all other elements of S are its *interior.* We call S *unrepairable* if one cannot make the array S satisfy the property $\mathcal{P}$ without changing the value of at least one element in its boundary. Otherwise, S is *repairable.* Observe the following simple facts.

- It suffices to only query the boundary elements of S in order to determine whether S is unrepairable. Moreover, if S is unrepairable, then A does not satisfy $\mathcal{P}$.
- If S is repairable, then we can delete all forbidden patterns from S by modifying only entries in its interior, without creating any new copies of forbidden patterns in A.

We call the process of understanding whether S is unrepairable using only its boundary elements *inference.* Note that the inference step does not make any additional queries.

A simple sublinear test A first attempt at a generic test for local properties is the following: we query $\Theta(\sqrt{n})$ intervals in $[n]$, each containing exactly $k-1$ consecutive elements, including the intervals $\{1, \dots, k-1\}$ and $\{n-k+2, \dots, n\}$, where the distance between each two neighboring intervals is $\Theta(\sqrt{n})$. A *block* is a subarray consisting of all elements in a pair of neighboring intervals and all elements between them. The crucial observation is that at least one of the following must be true, for any array A that is ε-far from $\mathcal{P}$ (recall that ε is fixed).

- At least one of the blocks is unrepairable.
- At least $\Omega(\sqrt{n})$ of the blocks do not satisfy $\mathcal{P}$.

Indeed, if the first condition does not hold, then one can make A satisfy $\mathcal{P}$ by only changing elements in the interiors of blocks that do not satisfy $\mathcal{P}$. Seeing that A is ε-far from $\mathcal{P}$ and that we do not need to modify elements in the interiors of blocks that satisfy $\mathcal{P}$, this implies that at least $\Omega(\sqrt{n})$ of the blocks do not satisfy $\mathcal{P}$.

Now we are ready to present the test: We query all $O(k\sqrt{n})$ elements of all intervals, and additionally, all $O(\sqrt{n})$ elements of $O(1)$ blocks. Querying all elements of all intervals suffices to determine (with probability 1) whether one of the blocks is unrepairable. If A is ε-far from $\mathcal{P}$ and does not contain unrepairable blocks, querying $O(1)$ full blocks will catch at least one block not satisfying $\mathcal{P}$ with constant probability, as desired. For more details, see Section 6.3 and the preliminary Section 6.2 that prepares the required infrastructure.

The optimal test Improving the query complexity requires us to construct a system of *grids* – which are merely subsets of $[n]$ – inspired by the behavior of binary search. In comparison, the approach of the previous test is essentially to work with a single grid. The first (and coarsest) grid contains only the first $k-1$ elements and the last $k-1$ elements of $[n]$. In other words, it is equal to $\{1, \ldots, k-1, n-k+2, \ldots, n\}$. The second grid refines the first grid – that is, it contains all elements of the first grid – and additionally, it contains $k-1$ consecutive elements whose center is $n/2$ (whenever needed, rounding can be done rather arbitrarily). We continue with the construction of grids recursively: To construct grid number $i+1$, we take grid number i and add $k-1$ elements in the middle of each block of grid i (blocks are defined as before). Note that the length of blocks is roughly halved with each iteration. We stop the recursive construction when the length of all of the blocks becomes no bigger than ck, where $c \geq 2$ is an absolute constant.

For each block B in grid number $i > 1$, we define its *parent*, denoted $\mathrm{Par}(B)$, as the unique block in interval $i-1$ containing it. A block B in the system of grid is *maximally unrepairable* if it is unrepairable, and all blocks (of all grids in the system) strictly containing it are repairable. It is not hard to see that different maximally unrepairable blocks have disjoint interiors.

The main observation now is that in order to make A satisfy $\mathcal{P}$, it suffices to only modify entries in the interiors of parents of maximally unrepairable blocks. If A is ε-far from $\mathcal{P}$, then the total length of these parents must therefore be $\Omega(n)$ (for constant ε). However, since the length of $\mathrm{Par}(B)$ is roughly twice the length of B, we conclude that the total length of all maximally unrepairable blocks is $\Omega(n)$. With this in hand, it can be verified that the following test has constant success probability. For each grid in the system, we pick one block of the grid uniformly at random, and query all entries of its boundary. Additionally, for the finest grid (whose block length is $O(k)$), we also query all interior elements of the picked block.

For more details, see Section 6.4 (which builds on the infrastructure of Section 6.2).

Running time in 1D We now show that the running time of the inference step for a block of length m is $m|\Sigma|^{O(k)}$. Summing over all block lengths, this would imply that the total running time of the test is $n|\Sigma|^{O(k)}$. The proof uses dynamic programming. Let S be an array of length m over Σ, and assume that $S(1), S(2), \ldots, S(k-1)$ and $S(m-k+2), \ldots, S(m)$ are all known. For each "level" from 1 to $m-k+1$, we keep a Boolean predicate for each of the $|\Sigma|^k$ possible patterns of length k over Σ. These predicates are calculated as follows.

- In the first level, the predicate of $\sigma = (\sigma_1, \ldots, \sigma_k)$ evaluates to TRUE if $S(1) = \sigma_1, \ldots, S(k-1) = \sigma_{k-1}$, and additionally, $\sigma \notin \mathcal{F}$, that is, σ is not a forbidden pattern. Otherwise, the predicate of σ is set to FALSE.

- For $i = 2$ to $m-k+1$, the predicate of $\sigma = (\sigma_1, \ldots, \sigma_k)$ in level i evaluates to TRUE if and only if $\sigma \notin \mathcal{F}$ and there exists $\sigma' = (\sigma'_0, \sigma_1, \ldots, \sigma_{k-1})$ that evaluates to TRUE in level $i-1$.

- Finally, the predicates in level $m-k+1$ are modified as follows: for all $\sigma = (\sigma_1, \dots, \sigma_k)$ so that $\sigma_j \neq S(m-k+j)$ for some $j \geq 2$, we set the predicate of σ to FALSE.

It is not hard to see that S is unrepairable if and only if all predicates at level $m-k+1$ are FALSE. The running time is $O(m|\Sigma|^{cd})$ for a suitable constant $c > 0$.

Generalization to higher dimensions The generalization to higher dimensions is relatively straightforward; the main difference is that the boundary of blocks now is much larger: blocks of size $m \times \ldots \times m$ have boundary of size $O(kdm^{d-1})$. Thus, essentially the same proof as above (with suitable adaptations of the definitions) yields a test with query complexity $O(kdn^{d-1})$ for constant ε. For the running time, we can no longer use dynamic programming; using the naive approach of enumerating over all possible interior elements of a block, we get that the inference time for a block of size $m \times \ldots \times m$ is $|\Sigma|^{O(m^d)}$, making the total running time of the test $|\Sigma|^{O(n^d)}$.

6.1.4 Other Related Work

Hyperfiniteness A graph is hyperfinite if, roughly speaking, it can be decomposed into constant size connected components by deleting only a small constant fraction of the edges. Newman and Sohler [110] investigated the problem of testing in hyperfinite graphs, showing that any property of hyperfinite bounded degree graphs is testable with a constant number of queries. While the graph with which we (implicitly) work – the hypergrid graph, whose vertices are in $[n]^d$ and two vertices are neighbors if they differ by 1 in one coordinate – is a hyperfinite bounded degree graph (for constant d), the results of [110] are incomparable to ours. Indeed, in our case the vertices are inherently ordered, and it does not make sense to allow adding edges between vertices that are not neighbors (as entries of $[n]^d$), unlike the case in [110], where one may add or remove edges arbitrarily between any two vertices. Still, the hyperfiniteness of our graph seems to serve as a major reason that local properties have sublinear tests.

Block tests for image properties The works of Berman, Murzabulatov and Raskhodnikova [29, 30] and the results in the next chapter, on testing of image properties (that is, on visual properties of 2D arrays), show that tests based on querying large consecutive blocks are useful for image property testing. Here, the general queries we make are quite different: we query the *boundaries* of blocks of different sizes, so the queries are *spherical*, in the sense that a block can be seen as a ball in the L_∞-metric on vectors in $[n]^d$, while its boundary can be be seen as the (width-k) sphere surrounding this ball. This introduces a new type of queries shown to be useful for image property testing.

6.1.5 Discussion and Open Questions

Small alphabets The results here are alphabet independent, and in particular, they work for alphabets over any size. An intriguing direction of research is to understand whether one can obtain more efficient general testability results for local properties of multi-dimensional arrays over smaller alphabets; this line of research has been conducted for specific properties of interest, like monotonicity and convexity [18, 111]. Note that the one-sided non-adaptive lower bound we prove here can be adapted to yield a $|\Sigma|^{\Omega(1)}$ lower bound for testing local properties over alphabets Σ of size smaller than n^d. The most interesting special case is that of constant-sized (and in particular, binary) alphabets. Here, no lower bounds that depend on n are known. For the case $d = 1$, it is known that all $O(1)$-local properties are constant query testable; this follows from a result of Alon et al. [10], who showed that any regular language is constant-query testable. However, it is not known whether an analogous statement holds in higher dimensions. That is, for any $d > 1$, the question whether all k-local properties of $[n]^d$-arrays over $\{0, 1\}$ are ε-testable with query complexity that depends only on d, k, and ε, first raised in [26] (see also [4]), remains an intriguing open question. We believe that positive results in this front might also shed light on the question of obtaining more efficient inference for large classes of properties, especially over small alphabets.

Does adaptivity help? This work does not provide any lower bounds for adaptive tests, and it will be interesting to do so; previously investigated properties likes monotonicity yield an $\Omega(d \log n)$ lower bound [36, 46], and we believe that "data flow" type properties, somewhat similar to our lower bound constructions, can provide instances of 2-local properties that require at least n^c queries, for some constant $c \leq 1$, for the adaptive two-sided case.

However, it is not clear whether better lower bounds (even bounds of the type $\Omega(n^{1+c})$) exist. It will be very interesting to prove better upper and lower bounds for testing local properties. Our conjecture is that *any* 2-local property is testable in $n^{1+o(1)}g(d)$ queries (where $g(d)$ depends only on d), but proving a statement of this type might be very difficult.

Using the unrepairability framework in other contexts We show that the concept of unrepairability allows to unify and reprove many property testing results on one-dimensional arrays. What about multi-dimensional arrays? for example, can one generalize the currently known proofs for "bounded derivative" properties (including monotonicity and Lipschitz continuity) in d dimensions to a larger class of local properties?

Inference As mentioned in Section 6.1.3, our test queries boundaries of block-like structures, and later infers whether each block is *unrepairable* (recall the definition from Section 6.1.3). The inference takes place without making any additional queries, and is based only on the property $\mathcal{P}$, the alphabet Σ, and the values of A in the boundary of the block.

The running time of the inference step is very large in general (although, as we have seen, in the 1D case it can be significantly improved using dynamic programming). The naive way to run the inference is by enumerating over all possible ways to fill the interior of the block, and checking whether each such possibility is indeed $\mathcal{F}$-free. The running time of this method is of order $|\Sigma|^{O(n^d)}$ in general for $d > 1$, and is exponential in n even if $|\Sigma| = 2$.

However, for many natural properties, inference can be done much more efficiently. For example, in monotonicity testing, the inference amounts to checking that no pair of boundary entries violates the monotonicity.Thus, we believe that understanding inference better – including tasks such as characterizing properties in which inference can be done efficiently, and understanding the inference time of specific properties of interest – would be an interesting direction for future research.

Organization Section 6.2 is devoted to the infrastructure needed for the proof, Section 6.3 presents a simple but non-optimal test, and finally, Section 6.4 presents the optimal test and proves Theorems 6.1 and 6.2. Matching lower bounds are proved in the full paper describing the results in this chapter [21].

6.2 The Grid Structure

In this section we present the grid-like structure in $[n]^d$ that we utilize for our tests.

Definition 6.3 (Interval partition)**.** *A subset $I \subseteq [n]$ is an* interval *if its elements are consecutive, that is, if $I = \{x, x+1, \ldots, x+y\}$ for some $x \in [n]$ and $y \geq 0$. For any $\ell \geq 0$, we denote the set of the smallest ℓ elements of I by $I[:\ell]$ and also define $I[\ell+1:\,] = I \setminus I[:\ell]$. In the degenerate case that $|I| < \ell$, we define $I[:\ell]$ to be equal to I.*

For $1 \leq w \leq n$, an (n,w)-interval partition *is a partition of $[n]$ into a collection of disjoint intervals $\mathcal{I} = (I_1, \ldots, I_t)$ where the number of elements in each interval I_i is either w or $w+1$, and for any $i < j$, all elements of I_i are smaller than those in I_j.*

Lemma 6.4. *For any positive integer n and $0 \leq i \leq \log n$, there exists an $(n, \lfloor n/2^i \rfloor)$-interval partition $\mathcal{I}_i$ containing exactly 2^i intervals, so that the family $\{\mathcal{I}\}_{i=0}^{\lfloor \log n \rfloor}$ satisfies the following. For any $i > j$ and interval $I \in \mathcal{I}_i$, there exists an interval $I' \in \mathcal{I}_j$ satisfying $I \subseteq I'$.*

Proof. For any i define $n_i = \lfloor n/2^i \rfloor$; observe that $n_0 = n$ and $n_{i+1} = \lfloor n_i/2 \rfloor$ for any i. We prove the lemma by induction on i, starting by defining $\mathcal{I}_0 = ([n])$. Given $\mathcal{I}_i = (I^i_1, \ldots, I^i_{2^i})$ in which all intervals are of length n_i or $n_i + 1$, we define $\mathcal{I}_{i+1}$ as follows. Each $I^i_j \in \mathcal{I}_i$, is decomposed into two intervals $I^{i+1}_{2j-1}, I^{i+1}_{2j}$ where $|I^{i+1}_{2j-1}|, |I^{i+1}_{2j}| \in \{n_{i+1}, n_{i+1}+1\}$, and all elements of I^{i+1}_{2j-1} are smaller than all elements of I^{i+1}_{2j}; observe that such a decomposition is indeed always possible. Now define $\mathcal{I}_{i+1} = (I^{i+1}_1, \ldots, I^{i+1}_{2^{i+1}})$. Clearly, the intervals of $\mathcal{I}_{i+1}$ satisfy the last condition of the lemma. □

In particular, we conclude that for any positive integer w and any $n \geq w$ there exists an integer $w/2 \leq w' \leq w$ for which an (n, w')-interval partition exists.

Definition 6.5 ((n,d,k,w)-grid)**.** *Let $2 \le w \le n$ be integers for which an (n,w)-interval partition $\mathcal{I} = (I_1,\dots,I_t)$ exists. For integers $2 \le k \le w$ and $d \ge 1$, the (d-dimensional)* (n,d,k,w)-grid *induced by $\mathcal{I}$ is the set*

$$G = \left\{ (x_1,\dots,x_d) \in [n]^d \;\middle|\; \exists i \in [d] \quad \text{such that } x_i \in \bigcup_{j=1}^{t} I_j[:k-1] \right\}.$$

We denote the family of all (n,d,k,w)-grids by $\mathcal{G}(n,d,k,w)$. As we have seen in Lemma 6.4, the family $\mathcal{G}(n,d,k,w)$ is non-empty for any $w = \lfloor n/2^i \rfloor$ satisfying $w \ge k$.

Definition 6.6 (G-block, Boundary, Closure)**.** *Two tuples $x = (x_1,\dots,x_d)$ and $y = (y_1,\dots,y_d)$ in $[n]^d$ are considered* neighbors *if $\sum_{i=1}^{d} |x_i - y_i| = 1$. Given a grid $G \in \mathcal{G}(n,d,k,w)$, consider the neighborhood graph of non-grid entries, i.e., the graph whose set of vertices is $V = [n]^d \setminus G$ and two entries are connected if they are neighbors. A G-*block *B is a connected component of this graph, and the* closure *of B is*

$$\overline{B} = \left\{ (x_1,\dots,x_d) \in [n]^d \;\middle|\; \exists (y_1,\dots,y_d) \in B \text{ such that } \forall i \in [d]\ |x_i - y_i| < k \right\}.$$

Note that $B \subseteq \overline{B}$. Define the boundary *of the block B as $\partial B = \overline{B} \setminus B$.*

The above notions can naturally be defined with Cartesian products. Recall that the Cartesian product of sets $X_1,\dots,X_d$, denoted $\prod_{j=1}^{d} X_j$ or $X_1 \times \ldots \times X_j$, is the set of all tuples $(x_1,\dots,x_d)$ with $x_j \in X_j$ for any $j \in [d]$. Let $G \in \mathcal{G}(n,d,k,w)$ be the grid induced by the interval partition $\mathcal{I} = (I_1,\dots,I_t)$. It is not difficult to verify that any G-block B can be defined as a Cartesian product $B = \prod_{j=1}^{d} I_{i_j}[k:\,]$ for some intervals $I_{i_1},\dots,I_{i_d} \in \mathcal{I}$ (not necessarily different).

$\overline{B}$ and ∂B can also be defined accordingly, as we detail next. For k as above, define $\overline{I_i} = I_i \cup I_{i+1}[:k-1]$ for any $1 \le i \le t$, where we take $I_{t+1} = \emptyset$ for consistency. Also define $\partial I_i = \overline{I_i} \setminus I_i[k:\,] = I_i[:k-1] \cup I_{i+1}[:k-1]$. With these in hand, we have

$$B = \prod_{j=1}^{d} I_{i_j}[k:\,]\ ; \qquad \overline{B} = \prod_{j=1}^{d} \overline{I_{i_j}}\ ; \qquad \partial B = \bigcup_{j=1}^{d} \overline{I_{i_1}} \times \ldots \times \overline{I_{i_{j-1}}} \times \partial I_{i_j} \times \overline{I_{i_{j+1}}} \times \ldots \times \overline{I_{i_d}} \tag{6.1}$$

Recall that $|I_{i_j}| \in \{w, w+1\}$ for any j, implying that $\left|I_{i_j}[k:]\right| \le w+2-k$ and $\left|\overline{I_{i_j}}\right| \le w+k$. Also note that $\left|\partial I_{i_j}\right| \le 2(k-1)$. Thus,

$$|B| \le (w+2-k)^d\ ; \qquad |\overline{B}| \le (w+k)^d\ ; \qquad |\partial B| \le 2d(k-1)\cdot(w+k)^{d-1}\ , \tag{6.2}$$

where the inequality on $|\partial B|$ holds since each set in the union expression in (6.1) is of size at most $(2k-2)(w+k)^{d-1}$. The following observation is a direct consequence of (6.1).

Observation 6.7. *Let $G \in \mathcal{G}(n,d,k,w)$. The boundary of any G-block is contained in G.*

Lemma 6.8. *For any $G \in \mathcal{G}(n,d,k,w)$, any width-$k$ subarray of an $[n]^d$-array intersects exactly one G-block B. Moreover, the subarray is contained in $\overline{B}$.*

Proof. Let $\mathcal{I} = (I_1, \dots, I_t)$ be the interval partition inducing G. Suppose that the subarray S is in location $(a_1, \dots, a_d)$ where $a_j \in I_{i_j}$ for some $i_1, \dots, i_d$ not necessarily distinct. In other words, the set of entries in S is $\prod_{j=1}^{d} S_j$ where $S_j = \{a_j, a_j + 1, \dots, a_j + k - 1\}$ for any $j \in [d]$. We argue that S is contained in $\overline{B}$, where $B = I_{i_1}[k\colon] \times \dots \times I_{i_d}[k\colon]$: The fact that $a_j \in I_{i_j}$ implies that $a_j + 1, \dots, a_j + k - 1 \in I_{i_j} \cup I_{i_j+1}[\colon k-1]$. It follows from (6.1) that $S \subseteq \overline{B}$. From Observation 6.7 we conclude that S does not intersect any block other than B, and it remains to show that S intersects B. Indeed, for any $1 \le j \le d$, the fact that $a_j \in I_{i_j}$ implies that one of the elements $a_j, \dots, a_j + k - 1$ must be contained in $I_{i_j}[k\colon]$. Denoting this element by b_j, we conclude that $(b_1, \dots, b_j) \in S \cap B$. □

6.3 Testing with Grid Queries

In this section we prove the following upper bound for all k-local properties; its proof serves as a warm-up towards proving the main upper bound of Theorem 6.1.

Theorem 6.9. *Any k-local property of $[n]^d$-arrays over any alphabet is ε-testable with one-sided error using no more than $2(d+1)n^{d-\frac{d}{d+1}}k^{\frac{d}{d+1}}\varepsilon^{-\frac{1}{d+1}}$ non-adaptive queries.*

The upper bound of Theorem 6.9 is sublinear in the size of the array as long as $k/\varepsilon^{1/d} = o(n)$. The rest of the section is dedicated to the proof of the theorem. We may assume that $k \le \varepsilon^{1/d}n/4$, as otherwise the expression in the statement of the theorem is larger than n^d and the proof follows trivially by querying all $[n]^d$ entries of the given input array. Under this assumption, it holds that $2k \le n^{d/(d+1)}k^{1/(d+1)}\varepsilon^{1/(d+1)}$.

Definition 6.10 (Unrepairable block)**.** *Let A be an $[n]^d$-array over Σ, and let $G \in \mathcal{G}(n,d,k,w)$. A G-block B is $(\mathcal{P}, A)$-*unrepairable *(or simply* unrepairable*, if $\mathcal{P}$ and A are clear from context) if any $[n]^d$-array A' over Σ that satisfies $A'(x) = A(x)$ for any $x \in \partial B$, including the case $A' = A$, contains an $\mathcal{F}$-copy in $\overline{B}$. Otherwise, the block B is said to be $(\mathcal{P}, A)$-*repairable.

Note that the (un)repairability of a block B is determined solely by the values of A on ∂B, and that an unrepairable block always contains an $\mathcal{F}$-copy. These two facts inspire the following lemma, which serves as the conceptual core behind the test of Theorem 6.9.

Lemma 6.11. *Suppose that A is an $[n]^d$-array that is ε-far from satisfying a k-local property $\mathcal{P}(\mathcal{F})$, and let $G \in \mathcal{G}(n,d,k,w)$ where $w \ge k$. Then at least one of the following holds.*

- *There exists a $(\mathcal{P}, A)$-unrepairable G-block.*
- *For at least an ε-fraction of the G-blocks B, there is an $\mathcal{F}$-copy in $\overline{B}$.*

Proof. Suppose that the first condition does not hold, that is, all G-blocks are $(\mathcal{P}, A)$-repairable. By Lemma 6.8, every $\mathcal{F}$-copy is contained in the closure of some G-block.

Let $\mathcal{C}$ denote the collection of all G-blocks B such that A contains an $\mathcal{F}$-copy in $\overline{B}$. By the repairability, the values of A in each block $B \in \mathcal{C}$ can be modified so that after the modification, A will not contain an $\mathcal{F}$-copy in $\overline{B}$. We stress that the modifications for each block B appear only in B itself and do not modify entries on the grid, so by Lemma 6.8, they cannot create new $\mathcal{F}$-copies in the closure of other blocks.

After applying all of the above modifications to A, we get an $\mathcal{F}$-free array, i.e., an array that satisfies $\mathcal{P}$. A was initially ε-far from $\mathcal{P}$, and the number of entries in each block is bounded by $(w + 2 - k)^d \leq w^d$, implying that at least an ε-fraction of the blocks belong to $\mathcal{C}$. □

Proof of Theorem 6.9. We may assume that $k^d/\varepsilon \leq n^d/2$, otherwise our test may trivially query all n^d entries of A. Our (non-adaptive) test T picks $W = \lfloor n^{d/(d+1)} k^{1/(d+1)} \varepsilon^{1/(d+1)} \rfloor \geq 2k$, and an integer w satisfying $k \leq W/2 \leq w \leq W$, for which an (n, w)-interval partition exists. T now makes the following queries.

1. T queries all entries of an arbitrarily chosen grid $G \in \mathcal{G}(n, d, k, w)$. The number of entries in any grid is at most $dn^d(k-1)/w \leq 2dn^{d-\frac{d}{d+1}} k^{\frac{d}{d+1}} \varepsilon^{-\frac{1}{d+1}}$.

2. T chooses a collection $\mathcal{B}$ of $2/\varepsilon$ G-blocks uniformly at random and queries all entries in these blocks. Since each block contains at most $(w + 2 - k)^d \leq W^d$ entries, the total number of queries is bounded by $2W^d/\varepsilon \leq 2n^{d-\frac{d}{d+1}} k^{\frac{d}{d+1}} \varepsilon^{-\frac{1}{d+1}}$. Note that the boundaries of all blocks are queried in the first step (since they are contained in the grid). Thus, for any block $B \in \mathcal{B}$, the test queries all entries of $\overline{B}$.

The total number of queries in the above two steps is $2(d+1)n^{d-\frac{d}{d+1}} k^{\frac{d}{d+1}} \varepsilon^{-\frac{1}{d+1}}$.

After querying all entries of the grid (and in particular, the whole boundaries of all of the blocks), T can determine for every G-block B whether it is $(\mathcal{P}, A)$-unrepairable or not. T rejects if at least one of the blocks is unrepairable or if it found an $\mathcal{F}$-copy in $\overline{B}$ for some $B \in \mathcal{B}$, and accepts otherwise. The test has one-sided error, since an unrepairable block must contain an $\mathcal{F}$-copy. In view of Lemma 6.11, T rejects arrays A that are ε-far from $\mathcal{P}$ with probability at least 2/3: If A satisfies the first condition of Lemma 6.11, then T always rejects. If the second condition holds, the probability that none of the $2/\varepsilon$ closures $\overline{B}$ for $B \in \mathcal{B}$ contains an $\mathcal{F}$-copy is bounded by $(1-\varepsilon)^{2/\varepsilon} < e^{-2}$, so T rejects with probability at least $1 - e^{-2} > 2/3$. □

6.4 Systems of Grids and Testing with Spherical Queries

In this section we prove Theorems 6.1 and 6.2. We do so by considering a system of grids with varying block sizes, defined as follows.

Definition 6.12. *Let $d > 0$ and $2 \leq k \leq w \leq n$ be integers. An (n, d, k, w)-*system of grids *is an $(r+1)$-tuple $(G_0, G_1, \ldots, G_r)$ of grids, for $r(n, w) = \lfloor \log(n/w) \rfloor$, so that*

- *$G_i \in \mathcal{G}(n, d, k, \lfloor n/2^{r-i} \rfloor)$ for any $0 \leq i \leq r$.*
- *$G_0 \supseteq G_1 \supseteq \ldots \supseteq G_r$ (as subsets of $[n]^d$). In particular, for any $i < j \leq r$, any G_i-block B is contained in a G_j-block B', and we say that B' is an* ancestor *of B. Specifically, the G_{i+1}-block containing B is called the* parent *of B and denoted by $Par(B)$. For the only G_r-block, B_r, we define $Par(B_r)$ as the whole domain $[n]^d$.*

$r(n, w)$ was chosen so that $w \leq n/2^r < 2w$, making G_0 a $\mathcal{G}(n, d, k, w')$-grid for $w \leq w' < 2w$. As we shall see, when working with such a system, unrepairability of blocks can be handled in a query-efficient way. The following lemma asserts that such a system of grids exists for any suitable choice of parameters.

Lemma 6.13. *An (n, d, k, w)-system of grids exists for all $d > 0$ and $2 \leq k \leq w \leq n$.*

Proof. Consider the family of interval partitions $\mathcal{I}_0, \ldots, \mathcal{I}_{\lfloor \log n \rfloor}$ obtained by Lemma 6.4. For each $0 \leq i \leq r(n, w)$ define G_i as the $(n, d, k, \lfloor n/2^{r-i} \rfloor)$-grid induced by $\mathcal{I}_{r-i}$. It is not hard to verify that $(G_0, \ldots, G_r)$ satisfies all requirements of an (n, d, k, w)-system of grids. □

For the rest of the section, fix a k-local property $\mathcal{P}(\mathcal{F})$ of $[n]^d$-arrays over Σ, and an $[n]^d$-array A over Σ. Consider an (n, d, k, w)-system of grids $(G_0, \ldots, G_r)$ constructed as described in the proof of Lemma 6.13, where w will be determined later. (For now it suffices to require, as usual, that $2 \leq k \leq w \leq n$.) We say that a G_i-block B is a $(\mathcal{P}, A)$-*witness* if one of the following holds.

- $i = 0$ and the array A contains an $\mathcal{F}$-copy in the closure $\overline{B}$.
- $i > 0$ and B is $(\mathcal{P}, A)$-unrepairable.

Recall that the closure of unrepairable blocks cannot be $\mathcal{F}$-free, so the closure of any witness block contains an $\mathcal{F}$-copy. We say that a witness block B is *maximal* if all of its ancestors are not witnesses, that is, they are repairable.

Observation 6.14. *Any $(\mathcal{P}, A)$-witness is contained in a maximal $(\mathcal{P}, A)$-witness.*

We define the *maximal witness family* $\mathcal{W}$ as the set of all maximal $(\mathcal{P}, A)$-witness blocks. Obviously, the blocks in $\mathcal{W}$ might come from different G_i's

Observation 6.15. $B_1 \cap B_2 = \emptyset$ *for any two blocks $B_1, B_2 \in \mathcal{W}$.*

Lemma 6.16. *All $\mathcal{F}$-copies in A are fully contained in $\bigcup_{B \in \mathcal{W}} \overline{B}$.*

Proof. Let F be an $\mathcal{F}$-copy in A. By Lemma 6.8, F is contained in the closure of a unique G_0-block B_F; hence, B_F is a $(\mathcal{P}, A)$-witness. From Observation 6.14 we have $B_F \subseteq B'$ for some maximal $(\mathcal{P}, A)$-witness B'. We conclude that $F \in \overline{B_F} \subseteq \overline{B'}$. □

Lemma 6.17. *One can make A satisfy $\mathcal{P}$ by only modifying entries of A in $\bigcup_{B\in\mathcal{W}} Par(B)$.*

Proof. Fix $B \in \mathcal{W}$. B is a maximal $(\mathcal{P}, A)$-witness, so $\mathrm{Par}(B)$ is repairable.[1] Thus, One can make $\overline{\mathrm{Par}(B)}$ $\mathcal{F}$-free by only modifying entries inside $\mathrm{Par}(B)$. By Lemma 6.8, width-k subarrays that are not fully contained in $\overline{\mathrm{Par}(B)}$ are left unchanged. Therefore, this modification does not create any new $\mathcal{F}$-copies in A. Seeing that all $\mathcal{F}$-copies in A are originally contained in $\bigcup_{B\in\mathcal{W}} \overline{B} \subseteq \bigcup_{B\in\mathcal{W}} \overline{\mathrm{Par}(B)}$, applying these modifications for all $B \in \mathcal{W}$ deletes all $\mathcal{F}$-copies in A without creating new ones, so in the end of the process A satisfies $\mathcal{P}$. $\square$

We may assume that $k \leq \varepsilon^{1/d}n/10$, as otherwise the expression in the theorem is $\Omega(n^d)$. We choose $w = 2k$, working with an $(n, d, k, 2k)$-system of grids from now on. A very useful consequence of this choice of w is that here the parent of a block B cannot be much larger than B itself.

Lemma 6.18. *Let $(G_0, G_1, \ldots, G_r)$ be an $(n, d, k, 2k)$-system of grids. Then for any $0 \leq i \leq r$ and any G_i-block B it holds that $|Par(B)|/|B| < 3^d$.*

Proof. For $i = r$ this is trivial. Now fix $i < r$ and let B be a G_i-block. Recall that, following (6.1), one can write $B = \prod_{j=1}^d I_{i_j}[k\colon]$ where each interval I_{i_j} (for $j \in [d]$) is of size at least $2k \geq 4$. On the other hand, we can also write $\mathrm{Par}(B) = \prod_{j=1}^d I'_{i'_j}[k\colon]$ where $I'_{i'_j} \supseteq I_{i_j}$ for any $j \in [d]$. It is not hard to verify that $|I'_{i'_j}| \leq 2|I_{i_j}| + 1$ most hold, and so

$$\frac{|\mathrm{Par}B|}{|B|} = \prod_{j=1}^{d} \frac{|I'_{i'_j}| - (k-1)}{|I_{i_j}| - (k-1)} \leq \prod_{j=1}^{d} \frac{2|I_{i_j}| + 1 - (k-1)}{|I_{i_j}| - (k-1)} \leq \left(\frac{2 \cdot 2k - k + 2}{2k - k + 1}\right)^d < 3^d$$

where the second inequality holds since $|I_{i_j}| \geq 2k$ for any j. $\square$

The next corollary follows immediately from Lemmas 6.17 and 6.18.

Corollary 6.19. *Suppose that A is ε-far from $\mathcal{P}$. Then the total number of entries in the blocks of $\mathcal{W}$ is at least $\varepsilon(n/3)^d$.*

We are now ready for the proof of the main upper bound of this chapter, Theorem 6.1.

Proof of Theorem 6.1

As before, we may assume that $k \leq \varepsilon^{1/d}n/10$. For larger k, the expression in the theorem dominates n^d and thus becomes trivial. Consider the $(n, d, k, 2k)$-system of grids $(G_0, G_1, \ldots, G_r)$ mentioned above. For any $0 \leq i \leq r$, define $\delta_i = |\mathcal{B}_i \cap \mathcal{W}|/|\mathcal{B}_i|$, where $\mathcal{B}_i$ is the set of all G_i-blocks. In other words, δ_i is the fraction of maximal witnesses among the G_i-blocks. By Corollary 6.19, if A is ε-far from $\mathcal{P}$ then $\sum_{i=0}^r \delta_i \geq \varepsilon/3^d$. Define $r' = \lfloor \log(\varepsilon^{1/d}n/k) \rfloor \geq 1$, noting that $G_{r'} \in \mathcal{G}(n, d, k, w_{r'})$ with $w_{r'} \geq 2k \cdot 2^{r'} \geq \varepsilon^{1/d}n$. Thus, the total number of blocks in $\mathcal{B}_{r'}$ is bounded by $(n/w_{r'})^d \leq 1/\varepsilon$.

[1]Note that when $B = B_r$ is the maximal witness considered, $\mathrm{Par}(B)$ is $[n]^d$; the latter is repairable for any non-empty property.

The test We iterate the following basic step $2 \cdot 3^d/\varepsilon$ times.

1. Pick $B \in \mathcal{B}_0$ uniformly at random and query all entries of $\overline{B}$.

2. For any $1 \le i \le r'$, pick $B \in \mathcal{B}_i$ uniformly at random and query all entries of $\bigcup_{B \in \mathcal{Q}_0} \partial B$.

Finally, the test rejects if and only if at least one of the blocks B picked during the process is a $(\mathcal{P}, A)$-witness. (Recall that querying all boundary entries of a G_i-block for $i > 0$ suffices to determine whether it is unrepairable, and thus a witness.) The test is clearly non-adaptive, and has one-sided error: It only rejects if it finds a witness. As we have seen earlier, all witnesses contain an $\mathcal{F}$-copy. The test is *canonical* in the following sense. The choice of queries in every basic step depends only on n, d, k, and (weakly) on ε, and is independent of the property $\mathcal{P}$ or the alphabet Σ. To determine which entries constitute a block, it suffices to know the parameters of the block, that depend only on n, d, k; the dependence in ε is only taken into account in the choice of r'. The test only considers $\mathcal{P}$ in order to determine whether each queried block is a witness.

Analysis Suppose that A is ε-far from $\mathcal{P}$. If $\delta_i > 0$ for some $i > r'$ then it must hold that $\delta_{r'} > 0$ as well (since any unrepairable G_i-block most contain an unrepairable $G_{i'}$-block for any $i' < i$). By the choice of r', we must have $\delta_{r'} \ge 1/|\mathcal{B}_{r'}| \ge \varepsilon$ in this case. If the above doesn't hold, then $\delta_i = 0$ for any $i \ge r'$, implying that $\sum_{i=0}^{r'-1} \delta_i \ge \varepsilon/3^d$. Therefore, in both cases, we have $\sum_{i=0}^{r'} \delta_i \ge \varepsilon/3^d$.

The probability that a random $\mathcal{B}_i$-block is a witness is at least δ_i, and therefore the probability that a single basic step leads to a rejection of A is at least $\sum_{i=0}^{r'} \delta_i \ge \varepsilon/3^d$. Running $2 \cdot 3^d/\varepsilon$ independent iterations of the basic step ensures that the test will accept A with probability at most $(1 - \varepsilon/3^d)^{2 \cdot 3^d/\varepsilon} \le e^{-2} < 2/3$, as desired.

Query complexity For $d = 1$, the query complexity of each basic step is $O(kr')$: The test queries $\overline{B}$ for a single block $B \in \mathcal{B}_0$, and the boundaries of r' larger blocks. Considering the parameters of our system of grids, we have $|B| \le 4k$ and so $|\overline{B}| < 6k$. On the other hand, the boundary of each of the larger blocks is of size at most $2k - 2$. Therefore, the total query complexity for the 1D test is $O(kr'/\varepsilon) = O\left(\frac{k}{\varepsilon} \log(\varepsilon n/k)\right)$ as desired. For $d > 1$, consider a single basic step, and for any $0 \le i \le r'$ let $B_i \in \mathcal{B}_i$ be the $\mathcal{G}_i$-block picked in this step. From (6.2) we have $|\overline{B_0}| \le (6k)^d$, while for any $i > 0$ we have $|\partial B_i| \le 2d(k-1)(4k \cdot 2^i + k)^{d-1} = O(d \cdot (4k)^d \cdot 2^{(d-1)i})$. Note that the last expression grows exponentially with $(d-1)i$, so the total number of queries in a single basic step is $O((6k)^d + d \cdot (4k)^d 2^{(d-1)r'})$. Plugging in r', we have $2^{(d-1)r'} = \frac{\varepsilon^{(d-1)/d}}{k^{d-1}} n^{d-1}$. As the test runs $O(3^d/\varepsilon)$ iterations of the basic step, we conclude that the total query complexity is bounded by $c^d k \varepsilon^{-1/d} n^{d-1}$ for an absolute constant $c > 0$, completing the proof of Theorem 6.1.

Proximity oblivious test The proof of Theorem 6.2 follows by a very simple modification of the proof of Theorem 6.1. The desired proximity oblivious test (POT) is the so called "basic step" from the above test, with r replacing r' (since r' depends on ε). The POT rejects if it infers that one of the blocks queried is a witness, like the above test. Its query complexity is $O(kr) = O(k \log n/k)$ for $d = 1$. In the case $d > 1$, the query complexity is dominated by the size of ∂B_r, which is bounded by $O(dkn^{d-1})$.

Clearly this POT has one-sided error, and its queries do not depend on the property $\mathcal{P}$ and the alphabet Σ (on the other hand, they do depend on n, d, k). Using the notation of the previous subsection and denoting by ε_A the Hamming distance of a given input A from $\mathcal{P}$, we get (exactly as in earlier parts of the proof) a rejection probability of at least $\sum_{i=0}^{r} \delta_i \geq \varepsilon_A/3^d$ for A, which is linear in ε_A for fixed d. This concludes the proof.

Chapter 7

Testing Meets Pattern Matching: The Modification Lemma

The results in this chapter appear in [26].

7.1 Introduction

Pattern matching is the algorithmic problem of finding occurrences of a fixed pattern in a given string. This problem appears in many settings and has applications in diverse domains such as computational biology, computer vision, natural language processing and web search. There has been extensive research concerned with developing algorithms that search for patterns in strings, resulting with a wide range of efficient algorithms [38, 58, 78, 98, 100]. Higher-dimensional analogues where one searches for a d-dimensional pattern in a d-dimensional array have received attention as well. For example, the 2D case arises in analyzing aerial photographs [14, 15] and the 3D case has applications in medical imaging. Given a string S of length n and a pattern P of length $k \leq n$, any algorithm which determines whether P occurs in S has running time $\Omega(n)$ [54, 117] and a linear lower bound carries over to higher dimensions. For the 2D and 3D case, when an $n \times n$ image is concerned, algorithms whose run time is $O(n^2)$ are known [15]. These algorithms have been generalized to the 3D case to yield running time of $O(n^3)$ [77]. Finally it is also known (e.g., [94]) that for the d-dimensional case it is possible to solve the pattern matching problem in time $O(d^2 n^d \log m)$ (where the pattern is an array of size m^d). It is natural to ask which tasks of this type can be performed in sublinear (namely $o(n^d)$) time for d-dimensional arrays.

Here, we are interested in deciding quickly whether a given d-dimensional array A is far from *not* containing a fixed d-dimensional pattern P. This is a special case of the setting discussed in the previous chapter, concerning local properties, where the forbidden family associated with the property contains a *single* forbidden pattern. For simplicity of presentation, all results in this chapter will be presented for cubic arrays in which $k_1 = \cdots = k_d$, but they generalize to non-cubic arrays in a straightforward manner. We consider the (tolerant) *pattern-freeness problem* where one needs to distinguish between the case that a given d-dimensional array A is ε_1-close to being P-free for a fixed pattern P, and the case that A is ε_2-far from being P-free, where $\varepsilon_1 < \varepsilon_2$. An $(\varepsilon_1, \varepsilon_2)$-test Q for this problem is a randomized algorithm that is given access to an array A, as well as its size and proximity parameters $0 \leq \varepsilon_1 < \varepsilon_2 < 1$. Q needs to distinguish with probability at least $2/3$ between the case that A is is ε_1-close to being P-free and the case that A is ε_2-far from being P-free. The query complexity of Q is the number of queries it makes in A.

Our interest in the pattern-freeness problem stems from several applications. In certain scenarios of interest, we might be interested in identifying quickly that an array is far from not containing a given pattern. For the one dimensional case, being far from not containing a given text may indicate a potential anomaly which requires attention (e.g., an offensive word in social network media), hence such testing algorithms may provide useful in anomaly detection. Many computer vision methods for classifying images are feature based: hence being far from containing a certain pattern associated with a feature may be useful in rejection methods that enable to quickly discard images that do not possess a certain visual property.

Beyond practical applications, devising property testing algorithms for the pattern freeness problem is of theoretical interest. In the first place, it leads to a combinatorial characterization of the distance from being P-free. Such a characterization has proved fruitful in graph property testing [7, 10] where celebrated graph removal lemmas were developed en route of devising algorithms for testing subgraph freeness. We encounter a similar phenomena in studying patterns and arrays: at the core of our approach for testing pattern freeness lies a *modification lemma* for patterns which we state next. We believe that this Lemma may be of independent interest and find applications beyond testing algorithms. Later we show one such application: computing the exact distance of a (one dimensional) string from being P-free can be done in linear time.

For a pattern P of size $k \times k \times \ldots \times k$, any of its entries that is in $\{0, k-1\} \times \ldots \times \{0, k-1\}$ is said to be a *corner* of P. We say that P is *almost homogeneous* if all of its entries but one are equal, and the different entry lies in a corner of P. Finally, P is *removable* (with respect to the alphabet Γ) if for any d-dimensional array A over Γ and any copy of P in A, one can destroy the copy by modifying one of its entries without creating new P-copies in A. The modification lemma states that for any d, and any large enough pattern P, when the alphabet is binary it holds that P is removable *if and only if* it is not almost homogeneous, and when the alphabet is not binary, P is removable provided that it is large enough.

Recent works [29, 30] have obtained tolerant tests for visual properties. As observed in [29, 30], tolerance is an attractive property for testing visual properties as real-world images are often noisy. With the modification lemma at hand, we show that when P is removable, the (relative) *hitting number* of P in A, which is the minimal size of a set of entries that intersects all P-copies in A divided by $|A|$, differs from the distance of A from P-freeness by a multiplicative factor that depends only on d (and not on P or A). This relation allows us to devise very fast $(5^{-d}\varepsilon, \varepsilon)$-tolerant tests for

P-freeness, as the hitting number of P in A can be well approximated using only a very small sample of blocks of entries from A. The query complexity of our test is $O(C_d/\varepsilon)$, where C_d is a positive constant depending only on the dimension d of the array. Note that our characterization in terms of the hitting number is crucial: merely building on the fact that A contains many occurrences of P (as can be derived directly from the modification lemma) and randomly sampling $O(1/\varepsilon)$ possible locations in A, checking whether the sub-array starting at these locations equals P would lead to query complexity of $O(k^d/\varepsilon)$. Note that our test is optimal (up to a multiplicative factor that depends on d), as any test for this problem makes $\Omega(1/\varepsilon)$ queries.

The one dimensional setting, where one seeks to determine quickly whether a string S is ε-far from being P-free is of particular interest. We are able to leverage the modification Lemma and show that the distance of a string S from being P-free for a fixed pattern P (that is not almost homogeneous) is *exactly equal* to the hitting number of P in A. For an arbitrary constant $0 < c < 1$, this characterization allows us to devise a $((1-c)\varepsilon, \varepsilon)$-tolerant test making $O_c(\varepsilon^{-1})$ queries for this case. For the case of almost homogeneous patterns, and an arbitrary constant $c > 0$, we devise a $((1/16 + c)\varepsilon, \varepsilon)$-tolerant test that makes $O_c(1/\varepsilon)$ queries. Whether tolerant tests exist for almost homogeneous patterns of dimension larger than 1 is an open question.

Moreover, the characterization via the hitting number implies an $O(n + k)$ algorithm that calculates (exactly) the distance of A from being P-free where P is an *arbitrary* pattern (that may be almost homogeneous). We are not aware of a previous algorithm for the distance computation problem. Unlike the one-dimensional case, in d dimensions we do not know of a clean combinatorial description of the distance to being P-free for higher dimension. Furthermore, it can be shown via a direct reduction from covering problems in the plane [72], that for dimension $d > 1$ there exists patterns P for which calculating the distance to P-freeness is NP-hard.

Related Work The problem of testing pattern freeness is related to the study of testing subgraph-freeness (see e.g. [3, 7] and Chapter 2). This line of work examines how one can test quickly whether a given graph G is H-free or ε-far from being H-free, where H is a fixed subgraph. In this problem, a graph is ε-far from being H-free if at least an ε-fraction of its edges and non-edges need to be altered in order to ensure that the resulting graph does not contain H as a (not necessarily induced) subgraph. A key component in these works are removal lemmas: typically such lemmas imply that if G is ε-far from being H-free, it necessarily contains a "large" number of copies of H. Perhaps the best example for this phenomena is the triangle removal lemma which asserts that for every $\varepsilon \in (0, 1)$, there exists $\delta = \delta(\varepsilon) > 0$ such that if an n-vertex graph G is ε-far from being triangle free, then G contains at least δn^3 triangles (see e.g., [13] and the reference within).

Alon et al. showed [10] that regular languages over $\{0, 1\}$ are strongly testable. Testing pattern-freeness (1-dimensional, binary alphabet, constant pattern length k) is a special case of the former, since the language of all strings avoiding a fixed pattern is regular. The query complexity of their test is $O\left(\frac{c}{\varepsilon} \cdot \ln^3(\frac{1}{\varepsilon})\right)$, where c is a constant that depends on the minimal size of a DFA A_L, that accepts the regular language L. It is shown in [10] that c can be taken to be $O(s^3)$ where s is the size of A_L. In the case of the regular language considered here a simple pumping-lemma inspired argument shows that $s \geq \Omega(k)$. Hence the upper bound on testing pattern freeness implied by their algorithm is $O\left(\frac{k^3}{\varepsilon} \cdot \ln^3(\frac{1}{\varepsilon})\right)$. Our 1D test solves a very restricted case of the problem the test of [10] deals with, but it achieves a better query complexity of $O(1/\varepsilon)$ in this setting. Moreover, our test is much simpler and can be applied in the more general high dimensional setting, or when the pattern length k is allowed to grow as a function of the string length n.

The problem of testing *submatrix freeness* was investigated in [6, 8, 68, 69, 71]. As opposed to our case, which is concerned with *tight* submatrices, all of these results deal with submatrices that are not necessarily tight (i.e. the rows and the columns need not be consecutive). Quantitatively, the submatrix case is very different from our case: in our case P-freeness can be testable using $O(\varepsilon^{-1})$ queries, while in the submatrix case, for a binary submatrix of size $k \times k$ a lower bound of $\varepsilon^{-\Omega(k^2)}$ on the needed number of queries is easy to obtain, and in the non-binary case there exist 2×2 matrices for which there exists a super polynomial lower bound of $\varepsilon^{\Omega(\log 1/\varepsilon)}$.

The 2D part of the results presented here adds to a growing literature concerned with testing properties of images [29, 115, 121]. Ideas and techniques from the property testing literature have recently been used in the fields of computer vision and pattern recognition [96, 99].

Notation and definitions We let $[[n]]$ denote the set $\{0, \ldots, n-1\}$. Here, for convenience, we view a d-dimensional (cubic) array A over an alphabet Γ is a function from $[[k]]^d$ to Γ. The $x = (x_1, \ldots, x_d)$ *entry* of A, denoted by A_x, is the value of the function A at location x. Let P be a (k, d)-array over an alphabet Γ of size at least two. We say that a d-dimensional array A *contains* a copy of P (or a P-copy) *starting in location* $x = (x_1, \ldots, x_d)$ if for any $y \in [[k]]^d$ we have $A_{x+y} = P_y$. Finally, A is P-free if it does not contain copies of P.

The absolute and relative distance to P-freeness will be denoted by $d_P(A)$ and $\delta_P(A)$, respectively. Here, $d_P(A)$ is the minimum absolute number of entry modifications required to make A free from P-copies, and $\delta_P(A) = d_P(A)/|A|$.

For an array A and a pattern P we will call a set of entries in A whose modification can turn it to be P-free a *deletion set* and therefore it is natural to call $d_P(A)$ (the absolute distance of A to P-freeness) the *deletion number*, since it is the size of a minimal deletion set. In a similar manner, for a given set of entries in A, if every P-copy in A contains at least one of these entries, we call it a *hitting set* and we call the size of a minimal hitting set the *hitting number*, denoted by $h_P(A)$. For all notations here and above, in the 1-dimensional case we will replace A by S (for String).

We recall several definitions regarding tolerant testability and distance estimation [113]. Let $\mathcal{P}$ be a property of arrays and let $h_1, h_2 : [0, 1] \to [0, 1]$ be two monotone increasing functions. An (h_1, h_2)-*distance approximation* algorithm for $\mathcal{P}$ is given query access to an unknown array A. The algorithm outputs an estimate $\widehat{\delta}$ to $\delta_P(A)$, such that with probability at least $2/3$ it holds that $h_1(\delta_P(A)) \leq \widehat{\delta} \leq h_2(\delta_P(A))$. Finally, for a property $\mathcal{P}$ and for $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$, an $(\varepsilon_1, \varepsilon_2)$-*tolerant* *test* for $\mathcal{P}$ is given query access to an array A. The test *accepts* with probability at least $2/3$ if A is ε_1-close to $\mathcal{P}$, and *rejects* with probability at least $2/3$ if A is ε_2-far from $\mathcal{P}$. Finally, we define the additive (multiplicative) *tolerance* of the test above as $\varepsilon_2 - \varepsilon_1$ ($\varepsilon_2/\varepsilon_1$ respectively).

Main Results The modification lemma result is central in the study of minimal deletion sets. It classifies the possible patterns into ones that are removable and ones that are not. The result that the vast majority of patterns are removable is used extensively throughout the chapter in the design and proofs of algorithms for efficient testing of pattern freeness (in one and higher dimensions) as well as for the exact computation of the deletion number in one-dimension. Our 1D modification lemma (Lemma 7.10) gives the following full characterization of one-dimensional patterns (i.e. strings). A binary pattern is removable *if and only if* it not almost homogeneous, while *any* pattern over a larger alphabet is removable. The multidimensional version of the lemma (Lemma 7.10) makes the exact same classification, but for (k,d)-arrays for which $k \geq 3 \cdot 2^d$. The fact that most patterns are removable is very important for analyzing the deletion number (which is the distance to pattern freeness). As an example, a simple observation is that a removable pattern appears at least $d_P(A)$ times (possibly with overlaps) in the array A, which implies an ε-test that can simply check for the presence of the pattern in $1/\varepsilon$ random locations in the array at a sample complexity of $O(k/\varepsilon)$.

Another important part of our results here makes explicit connections between the deletion number and the hitting number for both one and higher dimensions. These are needed in order to get improved tests (e.g. for getting rid of k in the sample complexity) in d-dimensions as well as for linear time computation of the distance (deletion number) in 1-dimension. For the 1D case we show that the deletion number $d_P(S)$ equals the hitting number $h_P(S)$, which leads to an exact computation of $d_P(S)$ in time $O(n+k)$ (Theorem 7.19) as well as a tolerant tests for Pattern Freeness: An $(\varepsilon_1, \varepsilon_2)$-tolerant test for any $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$ at a complexity of $O(\varepsilon_2^2/(\varepsilon_2 - \varepsilon_1)^3)$ (Theorem 7.20) as well as an $((1-\tau)\varepsilon, \varepsilon)$-tolerant test for a fixed $\tau > 0$ and any $0 < \varepsilon \leq 1$ at a complexity of $O(\varepsilon^{-1}\tau^{-3})$ (Corollary 7.21). For higher dimensions, we show (Lemma 7.15) that $h_P(A) \leq d_P(A) \leq \alpha_d h_P(A) \leq \alpha_d k^{-d}$, a bound that relates the hitting number $h_P(A)$ and the deletion number $d_P(A)$ through a constant $\alpha_d = 4^d + 2^d$ that depends only on the dimension d. This bound enables a $((1-\tau)^d \alpha_d^{-1}\varepsilon, \varepsilon)$-tolerant test making $C_\tau \varepsilon^{-1}$ queries, where $C_\tau = O(1/\tau^d(1-(1-\tau)^d)^2)$ (Theorem 7.23).

Our main results are summarized in Table 7.1. Additional results regarding almost homogeneous patterns are given in the full version [26].

7.2 Modification Lemma

Theorem 7.1 (Modification Lemma)**.** *Let $d > 1$ and let P be a (k,d)-array over the alphabet Γ where $k \geq 3 \cdot 2^d$.*

dim.	template type	deletion number computation	modification lemma	test tolerance	query complexity
1D	general	$O(n+k)$	removable for any k	$1/(1-\tau)$	$O(1/\varepsilon\tau^3)$
	almost homog.	$O(n+k)$	not removable for any k	$(16+c)$	α_c/ε
2^+D	general	NP-Hard	removable for $k > 3 \cdot 2^d$	$(1-\tau)^{-d}\alpha_d$	$\beta_{d,\tau}/\varepsilon$
	almost homog.	–	not removable for any k	–	–

Table 7.1: Summary of results. $0 < \tau < 1$ and $c > 0$ are arbitrary constants. α_c is a constant that depends only on c. $\beta_{d,\tau}$ is a constant that depends only on d and τ. 'modification lemma' specifies if patterns are classified as removable or not. the 'test tolerance' is multiplicative.

1. *If $|\Gamma| = 2$ then P is removable if and only if it is not almost homogeneous.*

2. *If $|\Gamma| \geq 3$ then P is removable.*

Remark 7.2. *Theorem 7.1 states that any large enough binary pattern which is not almost homogeneous is removable. The requirement that the pattern is large enough is crucial, as the $2 \times \ldots \times 2$ pattern P satisfying $P_x = 0$ for any $x = (x_1, \ldots, x_d)$ with $x_1 = 0$ and $P_x = 1$ otherwise is not removable even though it is not almost homogeneous. To see this, consider the following $4 \times \ldots \times 4$ array A: $M_x = 0$ if either $x_1 = 0$, or $x_1 = 1$ and $x_i \in \{1, 2\}$ for any $2 \leq i \leq d$, or $x_1 = 2$ and $x_i \in \{0, 3\}$ for some $2 \leq i \leq d$. For any other value of x, $M_x = 1$. Note that A contains a copy of P starting at $(1, \ldots, 1)$, but flipping any bit in this copy creates a new P-copy in A. Still, the size of the counterexample is only $2 \times \ldots \times 2$ while in the statement of Theorem 7.1, the dependence is exponential in d. It will be interesting to understand what is the correct order of magnitude of the dependence of k on d.*

Proof of Theorem 7.1. The second statement of the theorem can be easily derived from the first statement; If P does not contain all letters in Γ then it is clearly removable, as changing any of its entries to any of the missing letters cannot create new P-copies. Otherwise, we can reduce the problem to the binary case: let σ_1, σ_2 be the letters in Γ that appear the smallest number of times in P. Consider the following (k, d)-array P' over $\{0, 1\}$: $P'_x = 0$ if $P_x \in \{\sigma_1, \sigma_2\}$ and $P'_x = 1$ otherwise. Observe that P' is not almost homogeneous, implying that it is removable. It is not hard to verify now that P is removable as well.

In what follows, we will prove the first statement. If P is binary and almost homogeneous then it is not removable: Without loss of generality $P_{(0,\ldots,0)} = 1$ and $P_x = 0$ for any $x \neq (0, \ldots, 0)$. Consider a $(2k, d)$-array A such that $A_{(0,\ldots,0)} = A_{(1,\ldots,1)} = 1$ and $A = 0$ elsewhere. Clearly, modifying any bit of the P-copy starting at $(1, \ldots, 1)$ creates a new copy of P in A, so P is not removable.

The rest of the proof is dedicated to the other direction. Suppose that P is a binary (k, d)-array that is not removable. We would like to show that P must be almost homogeneous. As P is not removable, there exists a binary array A containing a copy of P that such that flipping any single bit in this copy creates a new copy of P in A. This copy of P will be called the *template* of P in A.

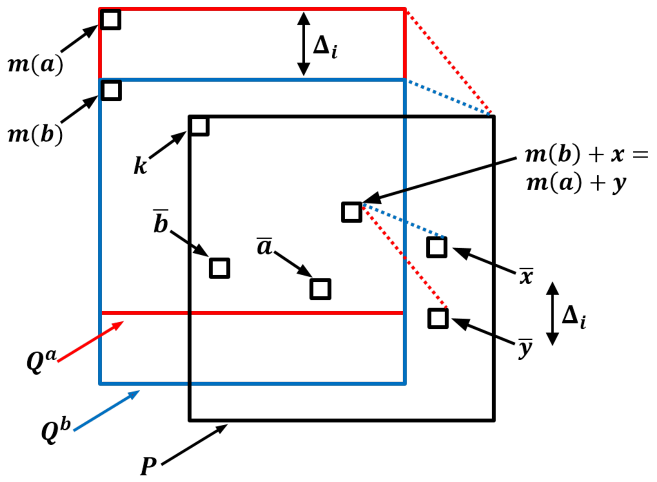


Figure 7.1: Illustration for Lemma 7.4. A 2-dimensional example, where i is the vertical coordinate: Flipping the bit (of the template P) at location $\bar{a}$ creates the P-copy Q^a at location $m(a)$. Similarly, the copy Q^b is created at location $m(b)$. Note that the pair of points $(\bar{x}, \bar{y})$ (which is (x, y) in P) and the copy locations pair $(m(a), m(b))$ are both (i, Δ_i)-related. The values P_x and P_y ($M_{\bar{x}}$ and $M_{\bar{y}}$) must be equal.

Clearly, all of the new copies created by flipping bits in the template must intersect the template, so we may assume that A is of size $(3k-2)^d$ and that the template starts in location $k = (k-1, \ldots, k-1)$.

For convenience, let $I = [[k]]^d$ denote the set of indices of P. For any $x \in I$ let $\bar{x} = x + k$; $\bar{x}$ is the location in A of bit x of the template.

Roughly speaking, our general strategy for the proof would be show that there exist at most two "special" entries in P such that when we flip a bit in the template, creating a new copy of P in A, the flipped bit usually plays the role of one of the special entries in the new copy. We will then show that in fact, there must be exactly one special entry, which must lie in a corner of P, and that all non-special entries are equal while the special entry is equal to their negation. This will finish the proof that P is almost homogeneous.

Definition 7.3. *Let $i \le d$ and let δ be positive integers. Let $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ be d-dimensional points. The pair (x, y) is* (i, δ)-related *if $y_i - x_i = \delta$ and $y_j = x_j$ for any $j \neq i$. An (i, δ)-related pair (x, y) is said to be an* (i, δ)-jump *in P if $P_x \neq P_y$.*

Lemma 7.4. *For any $1 \le i \le d$ there exists $0 < \Delta_i < k/3$ such that at most two of the (i, Δ_i)-related pairs of points from I are (i, Δ)-jumps in P.*

Proof. Recall that, by our assumption, flipping any of the $K = k^d$ bits of the template creates a new copy of P in A. Consider the following mapping $m : I \to [[2k-1]]^d$. $m(x_1, \ldots, x_d)$ is the starting location of a new copy of P created in A as a result of flipping bit $x = (x_1, \ldots, x_d)$ of the template (which is bit $\bar{x}$ of A). If more than one copy is created by this flip, then we choose the starting location of one of the copies arbitrarily.

Observe that m is injective, and let S be the image of m, where $|S| = K$. Let $1 \leq i \leq d$ and consider the collection of (one-dimensional) lines

$$\mathcal{L}_i = \big\{\{x_1\} \times \ldots \times \{x_{i-1}\} \times [[2k-1]] \times \{x_{i+1}\} \times \ldots \times \{x_d\} \quad | \quad \forall j \neq i : x_j \in [[2k-1]]\big\}.$$

Clearly $\sum_{\ell \in \mathcal{L}_i} |S \cap \ell| = K$. On the other hand, $|\mathcal{L}_i| = \prod_{j \neq i}(2k-1) < 2^{d-1} \prod_{j \neq i} k = 2^{d-1}K/k$, so there exists a line $\ell \in \mathcal{L}_i$ for which $|S \cap \ell| > k/2^{d-1} \geq 6$. Hence $|S \cap \ell| \geq 7$. Let $\alpha_1 < \ldots < \alpha_7$ be the smallest i-indices of elements in $S \cap \ell$. Since $\alpha_7 - \alpha_1 < 2k-1$ there exists some $1 \leq l \leq 6$ such that $\alpha_{l+1} - \alpha_l < k/3$. That is, S contains an (i, Δ_i)-related pair with $0 < \Delta_i < k/3$. In other words, there are two points $a, b \in I$ such that flipping $\bar{a}$ ($\bar{b}$) would create a new P-copy, denoted by Q^a (Q^b respectively), which starts in location $m(a)$ ($m(b)$ respectively) in A, and $(m(a), m(b))$ is an (i, Δ_i)-related pair. The following useful claim completes the proof of the lemma.

Claim 7.5. *For a and b as above, let (x, y) be a pair of points from I that are (i, Δ_i)-related and suppose that $y \neq \bar{a} - m(a)$ and that $x \neq \bar{b} - m(b)$. Then $P_x = P_y$.*

Proof. The bits that were flipped in A to create Q^a and Q^b are $\bar{a}, \bar{b}$ respectively. Since $y + m(a) \neq \bar{a}$, the copy Q_a contains the original entry of A in location $y + m(a)$. Therefore, $P_y = M_{y+m(a)}$ (as $M_{y+m(a)}$ is bit y of Q^a, which is a copy of P). Similarly, since $x + m(b) \neq \bar{b}$, we have $P_x = M_{x+m(b)}$. But since both pairs (x, y) and $(m(a), m(b))$ are (i, Δ_i)-related, we get that $m(b) - m(a) = y - x$, implying that $x + m(b) = y + m(a)$, and therefore $P_x = M_{x+m(b)} = M_{y+m(a)} = P_y$, as desired. $\square$

Clearly, the number of (i, Δ_i)-related pairs that do not satisfy the conditions of the claim is at most two, finishing the proof of Lemma 7.4. $\square$

Let $\Delta = (\Delta_1, \ldots, \Delta_d)$ where for any $1 \leq i \leq d$, we take Δ_i that satisfies the statement of Lemma 7.4 (its specific value will be determined later).

Definition 7.6. *Let $x \in I$. The set of* Δ-neighbors *of x is*

$$N_x = \big\{y \in I \quad \big| \quad \exists i : (x, y) \text{ is } (i, \Delta_i)\text{-related or } (y, x) \text{ is } (i, \Delta_i)\text{-related}\big\}$$

and the number of Δ-neighbors of x is $n_x = |N_x|$, where $d \leq n_x \leq 2d$. We say that x is a Δ-corner *if $n_x(\Delta) = d$ and that it is* Δ-internal *if $n_x(\Delta) = 2d$. Furthermore, x is* (Δ, P)-isolated *if $P_x \neq P_y$ for any $y \in N_x$, while it is* (Δ, P)-generic *if $P_x = P_y$ for any $y \in N_x$.*

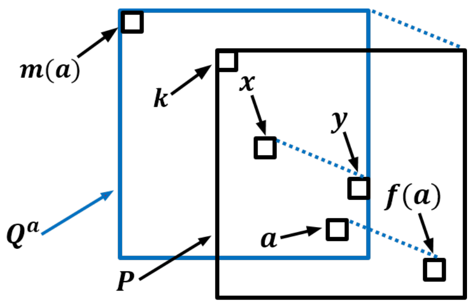
When using the above notation, we will sometimes omit the parameters (e.g. simply writing *isolated* instead of (Δ, P)-isolated) as the context is usually clear. The definition imposes a symmetric neighborhood relation, that is, $x \in N_y$ holds if and only if $y \in N_x$. If $x \in N_y$ we say that x and y are Δ-neighbors. Note that a point $x = (x_1, \ldots, x_d) \in I$ is a Δ-corner if $x_i < \Delta_i$ or $x_i \geq k - \Delta_i$ for any $1 \leq i \leq d$, and that x is Δ-internal if $\Delta_i \leq x_i < k - \Delta_i$ for any $1 \leq i \leq d$.

Claim 7.7. *Two (Δ, P)-isolated points in I cannot be Δ-neighbors.*

Proof. Suppose towards contradiction that $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ are two distinct (Δ, P)-isolated points and that (x, y) is (i, Δ_i)-related for some $1 \le i \le d$. Since $\Delta_i < k/3$, at least one of x or y participates in two different (i, Δ_i)-related pairs: if $x_i < k/3$ then $y_i + \Delta_i = x_i + 2\Delta_i < k$ so y is in two such pairs, and otherwise $x_i \ge \Delta_i$, meaning that x participates in two such pairs. Assume without loss of generality that the two (i, Δ_i)-related pairs are (t, x) and (x, y), then $P_t \ne P_x$ and $P_x \ne P_y$ as x is isolated. By Lemma 7.4, these are the only (i, Δ_i)-jumps in P.

Choose an arbitrary $j \ne i$ and take $v = (v_1, \ldots, v_d)$ where $v_j = \Delta_j$ and $v_l = 0$ for any $l \ne j$. Recall that $\Delta_j < k/3$, implying that either $x_j + v_j < k$ or $x_j - v_j \ge 0$. Without loss of generality assume the former, and let $x' = x + v$ and $y' = y + v$. Since x and y are (Δ, P)-isolated, and since $x' \in N_x$ and $y' \in N_y$, we get that $P_{x'} \ne P_x \ne P_y \ne P_{y'}$, and thus $P_{x'} \ne P_{y'}$ (as the alphabet is binary). Therefore, (x', y') is also an (i, Δ_i)-jump in P, a contradiction. □

Illustration for Definition 7.8. Recall that flipping a bit $\bar{a}$ in A creates a new P-copy Q^a (which contains $\bar{a}$), located at the point $m(a)$ in the coordinates of A. The bits x and a are *mapped* to y and $f(a)$ respectively.



Definition 7.8. *For three points $x, y, a \in I$, we say that x is* mapped *to y as a result of the flipping of a if $\bar{x} = m(a) + y$. Moreover, define the function $f : I \to I$ as follows: $f(x) = \bar{x} - m(x)$ is the location to which x is mapped as a result of flipping x.*

In other words, x is mapped to y as a result of flipping the bit a if bit $\bar{x}$ of A "plays the role" of bit y in the new P-copy Q_a that is created by flipping a. Note that

- If $\bar{x} - m(a) \notin I$ then x is not mapped to any point. However, this cannot hold when $x = a$, so the function f is well defined.
- For a fixed a, the mapping as a result of flipping a is linear: if x and y are mapped to x' and y' respectively, then $y - x = y' - x'$. In particular, if (x, y) is (i, Δ_i)-related for some $1 \le i \le d$ then (x', y') is also (i, Δ_i)-related.
- If x is mapped to y as a result of flipping a and $x \ne a$, then $P_x = P_y$.
- On the other hand, we always have $P_x \ne P_{f(x)}$.
- If x is Δ-internal and (Δ, P)-generic, then $f(x)$ must be (Δ, P)-isolated.

The first four statements are easy to verify. To verify the last one, suppose that x is internal and generic and let $z \in N_{f(x)}$; we will show that $P_{f(x)} \neq P_z$. Since x is internal, there exists $y \in N_x$ such that $y - x = z - f(x)$. Then y is mapped to z as a result of flipping x, since $\bar{y} = y + k = z + (x + k) - f(x) = z + \bar{x} - f(x) = z + m(x)$. Therefore $P_y = P_z$. On the other hand, $P_x = P_y$ as x is generic and $P_x \neq P_{f(x)}$, and we conclude that $P_z \neq P_{f(x)}$.

Lemma 7.9. *There is exactly one* (Δ, P)*-isolated point in* I*.*

Proof. Let $\mathcal{S}$ be the set of isolated points; our goal is to show that $|\mathcal{S}| = 1$. Consider the set

$$C = \{(x, y) : x, y \in I, (x, y) \text{ is an } (i, \Delta_i)\text{-jump for some } 1 \leq i \leq d\}.$$

Clearly, each point in $\mathcal{S}$ is contained in at least d pairs from C. By claim 7.7 no pair of isolated points are Δ-neighbors and therefore every pair in C contains at most one point from $\mathcal{S}$. By Lemma 7.4, $|C| \leq 2d$ which implies that $|\mathcal{S}| \leq 2$. On the other hand we have $|\mathcal{S}| \geq 1$. To see this, observe that the number of (Δ, P)-internal points in I is greater than $\prod_{i=1}^{d} k/3 \geq 2^{d^2}$, while the number of non-Δ-generic points is at most $2|C| \leq 4d$, implying that at least $2^{d^2} - 4d > 0$ of the internal points are generic. Therefore, pick an internal generic point $z \in I$. As we have seen before, $f(z)$ must be isolated. To complete the proof it remains to rule out the possibility that $|\mathcal{S}| = 2$. If two different (Δ, P)-isolated points $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$ exist, each of them must participate in exactly d pairs in C. This implies that both of them are Δ-corners with d neighbors. It follows that every Δ-internal point z must be generic (since an internal point and a corner point cannot be neighbors), implying that either $f(z) = a$ or $f(z) = b$.

Let $1 \leq i \leq d$ and define $\delta_i > 0$ to be the smallest integer such that there exists an (i, δ_i)-related pair (x, y) of generic internal points with $f(x) = f(y)$. For this choice of x and y we have $m(y) - m(x) = \bar{y} - f(y) - (\bar{x} - f(x)) = \bar{y} - \bar{x} = y - x$, so $(m(x), m(y))$ is also (i, δ_i)-related. In particular, we may take $\Delta_i = \delta_i$ (Recall that until now, we only used the fact that $\Delta_i < k/3$, without committing to a specific value). Without loss of generality we may assume that $f(x) = f(y) = a$. By Claim 7.5, any pair (s, t) of (i, Δ_i)-related points for which $s \neq \bar{y} - m(y) = f(y) = a$ and $t \neq \bar{x} - m(x) = f(x) = a$ is not an (i, Δ_i)-jump. Since b is not a Δ-neighbor of a, it does not participate in any (i, Δ_i)-jump, contradicting the fact that it is (Δ, P)-isolated. □

Finally, we are ready to show that P is almost homogeneous. Let $a = (a_1, \ldots, a_d)$ be the single (Δ, P)-isolated point in I. Consider the set

$$J = \{x = (x_1, \ldots, x_d) \in I : \Delta_i \leq x_i < \Delta_i + 2^d \text{ for any } 1 \leq i \leq d\}$$

and note that all points in J are Δ-internal. Let $1 \leq i \leq d$ and partition J into $(i, 1)$-related pairs of points. There are $2^{d^2-1} \geq 4d$ pairs in the partition. On the other hand, the number of non-generic points in J is at most $2|C| - (d - 1) < 4d$ (to see it, count the number of elements in pairs from C and recall that a is contained in at least d pairs). Therefore, there exists a pair (x, y) in the above partition such that x and y are both generic. As before, $f(x)$ and $f(y)$ must be isolated, and thus $f(x) = f(y) = a$, implying that $\Delta_i = \delta_i = 1$. We conclude that $\Delta = (1, \ldots, 1)$.

Claim 7.5 now implies that any pair (s, t) of $(i, 1)$-related points for which $s \neq \bar{y} - m(y) = f(y) = a$ and $t \neq \bar{x} - m(x) = f(x) = a$ is not an $(i, 1)$-jump. That is, for any two neighboring points $s, t \neq a$ in I, $P_s = P_t$, implying that $P_x = P_y$ for any $x, y \neq a$ (since $\Delta = (1, \ldots, 1)$, a Δ-neighbor is a neighbor in the usual sense). To see this, observe that for any two points $x, y \neq a$ there exists a path $x_0 x_1 \ldots x_t$ in I where x_j and x_{j+1} are neighbors for any $0 \leq j \leq t-1$, the endpoints are $x_0 = x$ and $x_t = y$, and $x_j \neq a$ for any $0 < j < t$. Since a is isolated, it is also true that $P_a \neq P_x$ for any $x \neq a$. To finish the proof that P is almost homogeneous, it remains to show that a is a corner. Suppose to the contrary that $0 < a_i < k-1$ for some $1 \leq i \leq d$ and let $b, c \in I$ be the unique points such that (a, b) and (c, a) are $(i, 1)$-related, respectively. Clearly $f(b) = a$, so a is mapped to $\bar{a} - m(b) = \bar{a} - \bar{b} + f(b) = c - a + a = c$ as a result of flipping b, which is a contradiction - as $P_a \neq P_c$ and $b \neq a, c$. This finishes the proof. $\square$

The above proof only works when the dimension is bigger than one, though it can be adapted to the one-dimensional case. However, we present here another proof for the one-dimensional case, which is simpler than the general proof above and works for any pattern which is not almost homogeneous (as opposed to the proof above, that required the forbidden pattern to also be large enough). The main strategy here is to consider the longest streaks of zeros and ones in the pattern - a strategy that cannot be used in higher dimensions.

Theorem 7.10 (1D Modification Lemma). *A one-dimensional pattern is removable if and only if it is almost homogeneous.*

Proof of Theorem 7.10. The reduction from a general alphabet to a binary one and the negative example for almost homogeneous patterns which were presented in the proof of Theorem 7.1 also hold here. It remains to prove that any 1-dimensional binary pattern that is not almost homogeneous is removable. Let $P = P_0 \ldots P_{k-1}$ be a binary pattern of length k, that is contained in an arbitrary binary string S. We need to show that one can flip one of the bits of P without creating a new P-copy in S. We assume that P contains both 0s and 1s (i.e. it is not homogeneous) otherwise flipping any bit would work. Therefore we can assume from now that $k \geq 3$ (since for $k = 1, 2$ all patterns are homogeneous or almost homogeneous).

Let us assume also that P starts with a 1, i.e. $P_0 = 1$ and let $t \leq k-1$ be the length of the longest 0-streak (sub-string of consecutive 0s) in P. Let $i > 0$ be the leftmost index in which such a 0-streak of length t begins. Clearly, $P_{i-1} = 1$ and $P_i = \ldots = P_{i+t-1} = 0$.

If $i + t \leq k$ (i.e. the streak is not at the end of P) then $P_{i+t} = 1$ and in such a case if we modify P_{i+t} to 0, the copy of P is removed without creating new P-copies in S. To see this, observe that a new copy cannot start at the bit flip location $i + t$ or within the 0-streak at any of its locations $i, \ldots, i + t - 1$ since the bits in these locations are 0 while the starting bit of P is 1. On the other hand, a new copy cannot start after $i+t$ since it must include the bit flip location or anywhere before P_i since otherwise it would contain a 0-streak of length $t+1$.

This implies that P contains exactly one 0-streak of length t at its last t locations. In particular, we have that at the last location $P_{k-1} = 1$, and if we denote by r the length of the longest 1-streak in P, a symmetric reasoning shows that P begins with its only longest 1-streak of length r.

If P is *not* of the form $1^s 0^t$, it can be verified that flipping P_s (the leftmost 0 in P) to 1 does not create any P-copy. The only case left is $P = 1^s 0^t$, where $s, t \geq 2$ since P is not almost homogeneous. Consider the bit of the string S that is to the left of P. If it is a 0 then we flip P_1 to 0 and otherwise, we flip P_0 to 0, where in both cases no new copy is created. □

7.3 Characterizations of the Deletion Number

We use the modification lemmas of Section 7.2 to investigate several combinatorial characterizations of the deletion number, which will in turn allow exact (and efficient) computations of the deletion number in the 1-dimensional case, as well as efficient approximation and testing of pattern freeness for removable patterns in the d-dimensional case for any d. In particular, we prove some surprising connections between minimal deletion sets and minimal hitting sets. The characterizations for almost homogeneous 1-dimensional patterns are given in the full version [26], along with an optimal algorithm to compute the exact deletion number and an optimal test for pattern freeness in that case. The rest of this section deals with removable patterns, for both the 1-dimensional and multi-dimensional settings. In the 1-dimensional case, we show that for any removable pattern there exist certain minimal hitting sets which are in fact minimal deletion sets. These are sets where none of the flips create new occurrences. Our constructive proof shows how to build such a set and allows for a linear time algorithm for finding the deletion number. The result is summarized in Theorem 7.11 and proved next.

Theorem 7.11 ($d_P(S)$ equals $h_P(S)$; Linear time computation of $d_P(S)$)**.** *For a binary string S of length n and a binary pattern P of length k that is removable, the deletion number $d_P(S)$ equals $h_P(S)$ and can be computed in time $O(n+k)$ and space $O(k)$.*

Proof. The main challenge is in proving that $d_P(S) = h_P(S)$, since then all we need is an algorithm that computes $h_P(S)$, which is relatively standard in template matching: Find the set $\mathcal{O}$ of all P-copies in S; Go though the P-copies in $\mathcal{O}$ from left to right, repeating the following: (i) Let P^* be the leftmost P-copy in $\mathcal{O}$; (ii) Increment the hitting set count by 1; (iii) Remove from $\mathcal{O}$ all the (following) P-copies that intersect P^* (those whose starting location is not to the right of the rightmost location in P^*);. Clearly, the complexity of the algorithm is dominated by the first step of finding $\mathcal{O}$, which can be done in $O(n+k)$ using, e.g., the KMP algorithm [98]. Taking the rightmost location in each of the visited P^*s creates a hitting set, which is minimal, due to the fact that the set of P^*s is independent.

It is trivial that $d_P(S) \geq h_P(S)$ and hence we have to show that $d_P(S) \leq h_P(S)$. Refer to Algorithm 1 below that constructs a set of bit flip locations. Note that the choice in Step 3 is possible using the modification lemma, while the choice in Step 4 is possible, since if h is contained in only one P-copy $P^0 \in \mathcal{D}$, by definition of $\mathcal{D}$ there is some $P^1 \in \mathcal{D}$ such that P^0 and P^1 intersect at some location x (in particular one of the 2 endpoints of P^0 must be in the intersection). Simply replace h by x. It is easy to verify that the set of locations $\mathcal{F}$ that it computes is a (particular) minimal hitting set of $\mathcal{O}$, and hence $|\mathcal{F}| = h_P(S)$. It is therefore sufficient to show that flipping the bit locations in $\mathcal{F}$ turns the string S to be P-free. This will be guaranteed, using the fact that $\mathcal{F}$ is a hitting set of $\mathcal{O}$, by Lemma 7.12 that shows that no bit flip of a location in $\mathcal{F}$ creates a new P-copy. Therefore, the proof of Lemma 7.12 will complete the proof of Theorem 7.11.

Algorithm 1

```
Input: Binary string $S$ of length $n$ and removable binary string $P$ of length $k$
Output: Minimal set $\mathcal{F}$ of flip locations in $S$ that make it $P$-free ($|F| = d_P(S)$)
  1. Find the set $\mathcal{O}$ of all $P$-copies in $S$
  2. Divide $\mathcal{O}$ into $\mathcal{I} \cup \mathcal{D}$, where $\mathcal{I}$ is the subset of $P$-copies that do not intersect any other
     $P$-copy in $\mathcal{O}$, while $\mathcal{D}$ is the subset of $P$-copies that intersect some other $P$-copy in $\mathcal{O}$.
  3. For each $P$-copy $P^* \in \mathcal{I}$ add to $\mathcal{F}$ a bit location whose flipping removes $P^*$ without creating
     any other $P$-copy
  4. Find a minimal hitting set $\mathcal{H}$ of $\mathcal{D}$ such that every location $h \in \mathcal{H}$ is contained in at least
     two $P$-copies in $\mathcal{D}$.
  5. Add $\mathcal{H}$ to $\mathcal{F}$
return $\mathcal{F}$
```

Lemma 7.12 (Flipping bits in F does not create new P-copies)**.** *Let $f \in \mathcal{F}$. Flipping the bit at location f does not create any new P-copy in S.*

Proof. Recall that $\mathcal{F}$ consisted of bits in $\mathcal{I}$ as well as bits in $\mathcal{D}$. Each of the bit flips that are in $\mathcal{I}$ was chosen (step 3 of Algorithm 1) using the modification lemma to be such that no new P-copy is created. The main challenge is in showing that the remaining bit flips, i.e. at locations $\mathcal{H}$, do not create any new P-copies. Notice our requirement that any location $h \in \mathcal{H}$ is contained in at least two P-copies. By symmetry considerations, we have the following:

Observation 7.13. [*Flipping an arbitrary bit in the intersection of 2 P-copies can create a new P-copy*] $\iff$ [*Flipping an arbitrary bit in a P-copy can create 2 new P-copies*]

By Observation 7.13, in order to show that bit flips in $\mathcal{H}$ do not create new P-copies, one can prove that an arbitrary bit-flip in a P-copy cannot create more than 1 P-copy. Applying Lemma 7.14 below thus completes the proof. $\square$

Lemma 7.14 (Any bit flip in a pattern P cannot create more than 1 new P-copy)**.** *Let $x \in [[k]]$. Flipping the bit P_x can create at most 1 new P-copy in S.*

Proof. The proof goes by contradiction, assuming that a bit flip in P has created two new P-copies P^1 and P^2, and will analyze separately the two possible cases:

Case 1: 'P^1 and P^2 intersect P from different sides'

In this case, flipping the bit location x of P creates a P-copy P^1 shifted t_1 locations to the left and a P-copy P^2 shifted t_2 locations to the right, where we assume w.l.o.g. that $t_1 < t_2$. One can verify that $P_{x-t_2} = P^2_{x-t_2} \neq P_x$ (and similarly that $P_x \neq P^1_{x+t_1} = P_{x+t_1}$). We assume that $P_x = 0$ and hence $P_{x-t_2} = 1$. See Figure 7.2 and its caption for the intuition of the proof.

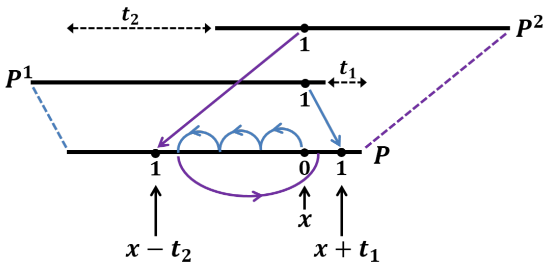


Figure 7.2: Illustration for Case 1: Our proof is based on 'skipping' along a 'path' from location x to location $x - t_2$ in P, while each skip is done between entries with equal values. A complete path from x to $x - t_2$ will give a contradiction, since $P_{x-t_2} \neq P_x$. The path starts at x and makes skips of size t_1 to the left as long as it does not pass $x - t_2$, then it makes a single skip to the right of size t_2. It repeats this traversal until reaching $x - t_2$.

Since the P-copy P^1 was created from P at a left offset of t_1 by the flipping at location x, we can infer that $P_y = P_{y+t_1}$ for any $y \in [[k - t_1]]$, $y \neq x$ (or informally that "P is t_1-cyclic except at x from the right"). Similarly, we know that "P is t_2-cyclic except at x from the left".

We define a 'path' of skips that starts from location x, makes skips of size t_1 to the left as long as it does not pass $x - t_2$, then it makes a single skip to the right of size t_2. Call this short path a traversal. The path repeats this traversal until reaching $x - t_2$. It is easy to verify that the path is always within the open range $(x - t_2, x + t_1)$ (except for the last step that reaches $x - t_2$). This implies in particular that the path does not go from $x + t_1$ to x or from x to $x - t_2$ (i.e. through the two only "value switching skips"), and hence the value of P along the path must be 0.

It remains to prove that the path eventually reaches $x - t_2$ and does not continue in some infinite loop. For each location y that the path goes through we can look at the value $y \pmod{t_1}$. Assume w.l.o.g. that for the 'target' location $x - t_2$ we have that $x - t_2 = 0 \pmod{t_1}$. This implies for the 'starting' location x that $x = t_2 \pmod{t_1} = 0$. Now, each skip by t_1 does not change the location $\pmod{t_1}$, while a skip by t_2 to the right increases the value by $t_2 \pmod{t_1}$. In other words, the sub-sequence of locations at the beginning of each traversal (before the first left skip) is of the form $\ell \cdot t_2 \pmod{t_1}$, for $\ell = 1, 2, 3, \ldots$. This is exactly the subgroup of $\mathbb{Z}_{t_1}$ (the additive group of integers modulo t_1) generated by the element t_2 and hence must contain the identity element $0 \pmod{t_1}$. This proves that the location $x - t_2$ will be reached.

Case 2: 'P^1 and P^2 intersect P from one (the same) side'

Flipping a location x in a P-copy P creates two new P-copies P^i $(i = 1, 2)$ that intersect P from the same side, w.l.o.g. right, at a shift of t_i, where $t_1 < t_2$. Refer to Figure 7.3 and its caption for the intuition of the proof. We 'follow' the two disjoint 'arrow paths' shown in the figure that lead from x in P to $x' := x - t_2$ in P^1 to reach a contradiction. Formally:

$$P_x = P^1_x = P^2_{x-t_2+t_1} = P_{x-t_2+t_1} = P^1_{x-t_2}$$

$$P_x \neq P^2_{x-t_2} = P_{x-t_2} = P^1_{x-t_2}$$

as desired. □

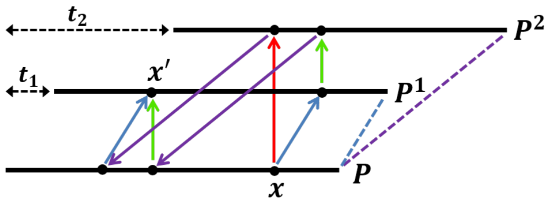


Figure 7.3: Illustration for Case 2: All arrows (ignoring directions) except the red one represent equality, while the red arrow represents inequality. The two disjoint 'arrow paths' from x in P to x' in P^1 imply that both $P_x = P_{x'}$ and $P_x \neq P_{x'}$, leading to contradiction.

The proof of the theorem now immediately follows from Lemma 7.12. □

For the multidimensional case, we start by showing that when P is removable, the hitting number $h_P(A)$ of A approximates the deletion number up to a multiplicative constant that depends only on the dimension d. This is done in two stages, the first of which involves the analysis of a procedure that proves the existence of a large collection of P-copies with small pairwise overlaps, among the large set of at least $d_P(A)$ P-copies that exist in A. This procedure heavily relies on the fact that P is removable. The second stage shows the existence of a large hitting set of the collection with small pairwise overlaps. The result is summarized in Lemma 7.15.

Lemma 7.15 (relation between distance and hitting number)**.** *Let P be a removable (k,d)-array over an alphabet Γ, and let A be an (n,d)-array over Γ. Let $\alpha_d = 4^d + 2^d$. It holds that: $h_P(A) \leq d_P(A) \leq \alpha_d h_P(A) \leq \alpha_d (n/k)^d$.*

Proof. The first inequality follows from the fact that one needs to modify at least one entry in any P-copy in A. For the third inequality, note that the set $\{(x_1,\dots,x_d) \in [[k]]^d : \forall 1 \leq i \leq d,\ x_i \equiv k-1 (\mathrm{mod}\ k)\}$ is a set of size $[[n/k]]^d$ that hits all $k \times \ldots \times k$ consecutive subarrays of A, and in particular all P-copies. It remains to prove that $d_P(A) \leq \alpha_d h_P(A)$. We may assume that the alphabet Γ is binary by applying the standard reduction from non-binary to binary alphabets presented in Section 7.2. We present a procedure on the array A that makes it P-free by sequentially flipping bits in it. In what follows, we will say that the *center* of a (k,d) matrix lies in location $(\lfloor k/2 \rfloor, \dots, \lfloor k/2 \rfloor)$ in the matrix. Let $\mathcal{P}$ be the set of all P-copies *before* A is modified. In Phase 1, the procedure "destroys" all P-copies in $\mathcal{P}$ by flipping central bits of a subset of the original P-copies in A, which is chosen in a greedy manner. However, these bit flips might create new P-copies in M, which are removed in Phase 2 using the modification lemma. The procedure maintains sets $\mathcal{A}, \mathcal{B}$ that contain the bits flipped in Phases 1,2 respectively.

- Let $\mathcal{P}$ be the set of all P-copies in A, $\mathcal{N} \leftarrow \phi$ $\mathcal{A} \leftarrow \phi$, $\mathcal{B} \leftarrow \phi$.
- **Phase 1:** While $\mathcal{P} \neq \phi$
  - Pick $Q \in \mathcal{P}$ arbitrarily.
  - Flip A_x where x is the center of Q.
  - Add Q to $\mathcal{A}$ and remove all P-copies containing x from $\mathcal{P}$.
  - Add all P-copies *created* by flipping A_x to $\mathcal{N}$.
- **Phase 2:** While $\mathcal{N} \neq \phi$
  - Pick $Q \in \mathcal{N}$ arbitrarily.
  - Pick a location x in Q whose flipping does not create new P-copies in A (exists by modification lemma).
  - Flip the bit A_x and add x to $\mathcal{B}$.

For the analysis of the procedure, we say that two P-copies Q, Q' in A whose starting points are $x = (x_1, \dots, x_d), y = (y_1, \dots, y_d) \in [[n]]^d$ respectively are 1/2-*independent* if $|x_i - y_i| \geq k/2$ for some $1 \leq i \leq d$. Note that 1/2-independence is a symmetric relation. A set of P-copies is 1/2-independent if all pairs of copies in it are 1/2-independent. Denote by $i_P(A)$ the maximal size of a 1/2-independent set in A, divided by n^d.

For Q and Q' as above, if Q' does not contain the center of Q then Q, Q' are 1/2-independent, as there is some $1 \leq i \leq d$ for which either $y_i < x_i + \lfloor k/2 \rfloor - (k-1) \leq x_i - k/2 + 1$ or $y_i > x_i + \lfloor k/2 \rfloor \geq$

$x_i + k/2 - 1$. In both cases $|y_i - x_i| \geq k/2$, implying the 1/2-independence. Therefore the set $\mathcal{A}$ generated by the procedure is 1/2-independent: if $Q, Q' \in \mathcal{A}$ are two different P-copies and Q was added to $\mathcal{A}$ before Q', then Q' does not contain the center of Q', so Q and Q' are 1/2-independent. Using the following claims, it is not hard finish the proof of the lemma.

Claim 7.16. *$i_P(A) \leq 2^d h_P(A)$.*

The proof of Claim 7.16 will be given later. For what follows, we say that P has a *cycle* of size $t = (t_1, \ldots, t_d) \in \mathbb{Z}^d$, if $P_x = P_y$ for every pair of locations $x = (x_1, \ldots, x_d), y = (y_1, \ldots, j_d) \in [[k]]^d$ such that $x_i \equiv y_i \pmod{|t_i|}\ \forall i \in [d]$. The following claim is straightforward to verify.

Claim 7.17. [Shifted occurrences imply a cyclic pattern] *If M contains two overlapping occurrences of A, at a relative offset of $t \in \mathbb{Z}^d$, then P has a cycle of size t.*

Claim 7.18. [Central bit flip creates few new occurrences] *Flipping the central bit of a P-occurrence in A creates at most 2^d new occurrences of P in A.*

We first show how to use these claims to finish the proof. Consider the sets $\mathcal{A}, \mathcal{B}$ after the procedure ends. The procedure flips $|\mathcal{A}| + |\mathcal{B}|$ bits in A, so $|\mathcal{A}| + |\mathcal{B}| \geq d_P(A) n^d$. On the other hand, $|\mathcal{A}| \leq i_P(A) n^d \leq (2n)^d h_P(A)$ as $\mathcal{A}$ is 1/2-independent. Claim 7.18 now implies that $|\mathcal{B}| \leq 2^d |\mathcal{A}|$, and we get that

$$n^d d_P(A) \leq |\mathcal{A}| + |\mathcal{B}| \leq (2^d + 1)|\mathcal{A}| \leq \alpha_d n^d h_P(A)$$

Dividing by n^d yields the desired inequality. We now prove the claims.

Proof of Claim 7.16. Let $\mathcal{S}$ be a 1/2-independent set of P-copies in A, which is of size $i_P(A)$. We will show that no point in $[[n]]^d$ is contained in more than 2^d copies from S, implying that to hit all copies of P in A (and in particular, all copies of P in $\mathcal{S}$) we will need at least $|\mathcal{S}|/2^d = i_P(A)/2^d$ entries. Suppose to the contrary that there are $2^d + 1$ copies from $\mathcal{S}$ that contain the point $x = (x_1, \ldots, x_d) \in [[n]]^d$. we will say that a copy from $\mathcal{S}$ containing x is i-lower if $k/2 \leq x_i - y_i < k$ and i-higher if $0 \leq x_i - y_i < k/2$ (note that $0 \leq x_i - y_i < k$ must hold). Therefore, there exist two copies $Q, Q' \in \mathcal{S}$ containing x, starting at $(y_1, \ldots, y_d)$ and $(y'_1, \ldots, y'_d)$ respectively, such that for any $1 \leq i \leq d$, Q is i-higher (i-lower) *if and only if* Q' is i-higher (i-lower respectively). But then, for any i, either $0 \leq y_i, y'_i < k/2$ or $k/2 \leq y_i, y'_i < k$, implying that $|y_i - y'_i| < k/2$, thus contradicting the fact that $\mathcal{S}$ is 1/2-independent. □

Proof of Claim 7.18. Assume that more than 2^d new occurrences are created. Since these occurrences overlap (at the bit flip location),the same argument as in Claim 7.16 implies that there must be two of them, P_1 and P_2, that are shifted (one from the other) by some vector $t \in \mathbb{Z}^d$, where $|t_i| < k/2\ \forall i \in [d]$. By Claim 7.17, P (and hence also P_1 and P_2) has a cycle of size t. Let x be the point in M of the (central) flipped bit in P_0 and consider the point $x' = x + t$, which is also in P_0, since $|t_i| < k/2\ \forall 1 \leq i \leq k$. The occurrence P_2 overlaps both locations x and x' (since both new occurrences P_1 and P_2 overlap the bit flip location x and P_2 is shifted by t from P_1, which overlaps

x). On one hand we have $M_x = M_{x'}$ (before the bit flip), since both locations belong to P_0, which has a cycle of size t. On the other hand, $M_x \neq M_{x'}$, since these locations both belong to P_2 and must be equal *after* flipping M_x as P_2 has a cycle length of t. This leads to a contradiction. □

The proof of the lemma is now complete by combining the above claims. □

7.4 Tests for Pattern Freeness

We describe efficient tests for both the one-dimensional and the d-dimensional removable patterns that have tolerance and query complexity that only depend on d (and not on k; using a completely naive test, it can be seen that the tolerance and the query complexity depend on k). The tests essentially approximate the hitting number, which is related to the deletion number by the characterizations that were shown in Section 7.3. We start by presenting the distance approximation algorithm for P-freeness, which has both additive and multiplicative errors.

Theorem 7.19 (Approximating the deletion number in one dimension)**.** *Let P be a removable string of length k and fix constants $0 < \tau < 1, 0 < \delta < 1/k$. Let $h_1, h_2 : [0,1] \to [0,1]$ be defined as $h_1(\varepsilon) = (1-\tau)\varepsilon - \delta$ and $h_2(\varepsilon) = \varepsilon + \delta$. There exists an (h_1, h_2)-distance approximation algorithm for P-freeness with query complexity and running time of $O(1/k\tau\delta^2)$.*

Note that $d_P(S) = h_P(S) \leq n/k$ always holds, so having an additive error parameter of $\delta \geq 1/k$ is pointless. The proof of Theorem 7.19 can be adapted to derive $(\varepsilon_1, \varepsilon_2)$-tolerant tests for any $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$, which we describe in Theorem 7.20.

Theorem 7.20. *Let P be a removable string of length k and let $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$. There exists an $(\varepsilon_1, \varepsilon_2)$-tolerant tester whose number of queries and running time are $O(\varepsilon_2^2/(\varepsilon_2 - \varepsilon_1)^3)$ where the constant term does not depend on k.*

Corollary 7.21 (Multiplicative tolerant test for pattern freeness in 1D)**.** *Fix $0 < \tau < 1$. For any $0 < \varepsilon \leq 1$ there exists a $((1-\tau)\varepsilon, \varepsilon)$-tolerant test whose number of queries and running time are $O(\varepsilon^{-1}\tau^{-3})$.*

It is not clear whether this upper bound is tight in general. However, for the important special case of tolerant testers with multiplicative tolerance of $1 + \tau$, where $\tau > 0$ is a constant, the above tester is optimal (up to a multiplicative constant that depends on τ), as is shown by taking $\varepsilon_2 = \varepsilon$ and $\varepsilon_1 = (1-\tau)\varepsilon$ in Theorem 7.20, leading to the multiplicative tester given in Corollary 7.21.

Proof of Theorems 7.19 and 7.20. Let S be a string of length $n \geq \beta k$, where $\beta = 3/\tau$. Write $\varepsilon = \delta_P(S)$ and let $H \subseteq [[n]]$ be a hitting set for P in S whose size is εn. That is, H is a minimal set of locations that satisfies the following: if S contains a copy of P starting at location l, then $\{l, \ldots, l+k-1\} \cap H \neq \phi$. For $i \in [[n]]$ let I_i denote the "cyclic interval" of length βk starting at i. That is, if $i + \beta k > n$ then $I_i = \{i, \ldots, n\} \cup \{0, \ldots, i + \beta k - n - 1\}$ and otherwise $I_i = \{i, \ldots, i + \beta_k - 1\}$.

Let the random variable X denote the size of the minimal hitting set H_i for P in the interval I_i, divided by βk, where $i \in [[n]]$ is chosen uniformly at random. Note that X is computable in time $O(\beta k)$, by Theorem 7.11. Let μ and σ^2 denote the expectation and the variance of X, respectively. By the minimality of H_i, we have that $|H_i| \leq |H \cap I_i|$ since the set in the RHS is a hitting set for P with respect to the interval I_i. Thus, $\mu \leq \mathbb{E}[|H \cap I_i|]/\beta k = \varepsilon$. Next we bound μ from below. Since H_i hits all P-copies that lie exclusively inside I_i, and by the minimality of H, we must have $|H_i| > |H \cap I'_i|$ where I'_i is the cyclic interval that starts in $i + k$ and ends in $(i + (\beta - 1)k - 1)$ mod n. Therefore, $\mu \geq \mathbb{E}[H \cap I'_i]/\beta k = (1 - 2/\beta)\varepsilon \geq (1 - \tau)\varepsilon$. To conclude, we have seen that $(1 - \tau)\varepsilon \leq \mu \leq \varepsilon$. To compute the variance of X, note that $0 \leq X \leq 1/k$, as there exist β entries in I_i such that any sub-interval of length k in I_i contains at least one of them. By convexity, the variance satisfies $\sigma^2 \leq k\mu(1/k - \mu)^2 + (1 - k\mu)(0 - \mu)^2 = \mu(1/k - \mu) \leq \varepsilon/k$. Now let $Y = \frac{1}{t}\sum_{j=1}^{t} X_j$ where the X_j are independent copies of X and t will be determined later. Then $\mathbb{E}[Y] = \mu$ and $\mathrm{Var}(Y) = \sigma^2/t \leq \varepsilon/kt$.

Recall that $(1 - \tau)\varepsilon \leq \mu \leq \varepsilon$, so to get the desired approximation, it suffices to estimate Y with an additive error of no more than δ with constant probability. Chebyshev's inequality implies that it suffices to have $Var(Y) = \Theta(\delta^2)$. In other words, it will be enough to sample $t = \Theta(\varepsilon/k\delta^2)$ blocks, each of size $\beta k = \Theta(k/\tau)$. In total, it is enough to make $\Theta(k\varepsilon(1/k - \varepsilon)/\tau\delta^2) = O(\varepsilon/\tau\delta^2)$ queries. In the setting of approximation, ε is not known in advance, but $\varepsilon \leq 1/k$ always holds, so sampling $t = \Theta(1/k^2\delta^2)$ blocks would suffice to get the desired additive error. The return value of the approximation algorithm will be its estimate of Y. The query complexity and running time are $\beta tk = \Theta(1/k\tau\delta^2)$. This finishes the proof of Theorem 7.19.

Now consider the setting of $(\varepsilon_1, \varepsilon_2)$-tolerant testing. By monotonicity of the tester, we can assume that we are given a string whose distance from P-freeness is either exactly ε_2 or exactly ε_1. Pick $\varepsilon = \varepsilon_2$, $\delta = (\varepsilon_2 - \varepsilon_1)/4$, $\tau = (\varepsilon_2 - \varepsilon_1)/4\varepsilon_2$, and sample $t = \Theta(\varepsilon/k\delta^2)$ blocks, with query complexity and running time $\Theta(\varepsilon/\tau\delta^2) = \Theta(\varepsilon_2^2/\varepsilon_2 - {\varepsilon_1}^3)$, as was stated above. If the given string S is ε_2-far from P-freeness, then with probability at least $2/3$, after sampling $t = \Theta(\varepsilon/\tau\delta^2)$ samples, the value of Y will be bigger than $(\varepsilon_2) * (1 - \tau) - \delta = (\varepsilon_2 + \varepsilon_1)/2$. On the other hand, if S is ε-close then with probability at least $2/3$, $Y \leq \varepsilon_1 + \delta < (\varepsilon_2 + \varepsilon_1)/2$ Therefore, the tester will answer that the input is ε_2-far if and only if $Y \geq (\varepsilon_2 + \varepsilon_1)/2$. This concludes the proof of Theorem 7.20. $\square$

For the multidimensional case, our distance approximation algorithm and tolerant test for P-freeness are given in Theorems 7.22 and 7.23. As their technical details are very similar to those in the 1D case, we provide only a sketch of the main ideas.

Theorem 7.22 (Approximating the deletion number in multidimensional arrays)**.** *Let P be a removable (k, d)-array and fix constants $0 < \tau \leq 1, 0 \leq \delta \leq 1/k^d$. Let $h_1, h_2 : [0, 1] \to [0, 1]$ be defined as $h_1(\varepsilon) = (1 - \tau)^d \alpha_d^{-1}\varepsilon - \delta$ and $h_2(\varepsilon) = \varepsilon + \delta$. There exists an (h_1, h_2)-distance approximation algorithm for P-freeness making at most $\gamma/k^d\tau^d\delta^2$ queries, where $\gamma > 0$ is an absolute constant, and has running time $\zeta_\tau/k^d\delta^2$ where ζ_τ is a constant depending only on τ.*

Theorem 7.23 (Multiplicative tolerant test for pattern freeness in multidimensional arrays)**.** *Fix $0 < \tau \leq 1$ and let P be a removable (k,d)-array. For any $0 < \varepsilon \leq 1$ there exists a $((1-\tau)^d \alpha_d^{-1}\varepsilon, \varepsilon)$-tolerant test making $C_\tau \varepsilon^{-1}$ queries, where $C_\tau = O(1/\tau^d(1-(1-\tau)^d)^2)$. The running time is $C'_\tau \varepsilon^{-1}$ where C'_τ depends only on τ.*

Proof sketch for Theorems 7.22 and 7.23. Take $\beta = 2/\tau$. Let A be an (n,d)-array where we may assume that $n \geq \beta k$ for a suitable choice of C. Again, the strategy is to take t (to be determined) independent samples of blocks of size $\beta k \times \ldots \times \beta k$ and compute the hitting number of each sampled block. Note that (as opposed to the one-dimensional case), computing the minimal hitting set is generally an NP-complete problem, but since the hitting number of each of these blocks is at most $\beta^d = \Theta(\tau^{-d})$, here we may compute it with running time that depends only on τ and d. As in the 1D case, the expected relative hitting number μ of a sampled block satisfies $(1-\tau)^d h_P(A) = (1-2/\beta)^d h_P(A) \leq \mu \leq h_P(A)$. The variance of the hitting number for a single sample is no bigger than $k^d(1/k^d - \mu)^2 + (1-k^d\mu)\mu^2 = \mu(1/k^d - \mu) \leq \mu/k^d$, so for t samples it is $O(h_P(A)/k^d t)$. To get additive error of at most δ with constant probability, we may have (by Chebyshev's inequality) $h_P(A)/k^d t = \Theta(\delta^2)$, or $t = \Theta(h_P(A)/k^d\delta^2)$.

Therefore, for an approximation algorithm (in which we don't know $h_P(A)$ in advance, though we have an upper bound of $h_P(A) \leq 1/k^d$), $t = \Theta(k^{-2d}\delta^{-2})$ sampled blocks are enough, and the total number of samples is $O(1/k^d\tau^d\delta^2)$. For a $((1-\tau)^d\varepsilon, \varepsilon)$-tolerant tester for the hitting number (which translates to a $((1-\tau)^d\alpha_d^{-1}\varepsilon, \varepsilon)$-tolerant tester for the deletion number), as observed in the 1D case, when deciding on the number of samples we may assume that $h_P(A) = \varepsilon$ and pick $\delta = \Theta((1-(1-\tau)^d)\varepsilon)$, so $t = \Theta(\varepsilon/k^d\delta^2) = \Theta(1/k^d(1-(1-\tau)^d)^2\varepsilon)$ sampled block suffice. Since each block is of size $\Theta(k^d/\tau^d)$, the total number of queries is $O(C_\tau\varepsilon^{-1})$ where $C_\tau = 1/\tau^d(1-(1-\tau)^d)^2$, while the running time is $C'_\tau\varepsilon^{-1}$, where C'_τ depends on the time required to compute the hitting number in a single sampled block. $\square$

7.5 Discussion and Open Questions

This chapter address the property of pattern-freeness for a single forbidden pattern. Naturally, the problem of *approximate* pattern matching is an intriguing venue for further research and might be of practical interest. While recent results by Chan et al. [49] (see also [134]) address a related problem in one dimension, efficient testing for approximate pattern matching in higher dimensions is left as an open problem. The family of forbidden patterns for this problem might consist of a pattern and all patterns that are close enough to it, and the distance measures between patterns might also differ from the Hamming distance (e.g., ℓ_1 distance for grey-scale patterns).

It is also desirable to settle the problem of testing pattern freeness for the almost homogeneous case by either finding an efficient test for the almost homogeneous multi dimensional case, or proving that an efficient test cannot exist for such patterns. Additionally, it is of interest to examine which of the $[[k]]^d$ patterns with $k < 3 \cdot 2^d$ are removable.

Chapter 8

Conclusions

In this thesis, we conduct a systematic study of property testing in data with complex structure, a topic that has not been well-understood throughout the years. The contributions span several new concepts and new types of results in multiple objects of interest, including:

Structural characterizations We prove several wide structural results that lead to new efficient property testing algorithms for very general classes of properties, including all hereditary properties of 2D structured objects (Chapter 2) and all local properties of one- and multi-dimensional arrays (Chapter 6). Such general testability results were known before only for objects that exhibit inherent symmetry, like unordered graph properties and symmetric distribution properties, and were missing for objects with complex structure. The characterizations obtained here can roughly be divided into two types.

The first type, exhibited in Chapter 2, substantially extends important characterizations from the unordered regime (i.e., for unordered graph properties) to the ordered regime by completely relaxing the need for symmetry in the combinatorial techniques and proofs. Informally, this type of extension suggests that "ultimately global" properties of structured objects behave in high-level quite similarly to symmetric properties of unordered objects. Following the results of Chapter 2, this was subsequently explored in recent years to obtain a relatively good understanding of what makes global properties testable in ordered structures [24, 25].

The second type, described in Chapter 6, is entirely new and does not resemble any characterizations from the unordered/symmetric setting. Instead, it takes a new perspective on property testing which relies on a new, seemingly useful, notion of locality discussed below.

What is testable? One of the central barriers to good understanding of structured property testing has been a lack of relevant notions and definitions. Indeed, to understand "what is testable", one needs to define classes of properties of interest, with a shared trait that allows for efficient testability. In Chapter 6, we define the class of *local* properties, and show that numerous interesting properties are local: monotonicity, Lipschitz continuity, convexity, submodularity, and others. Notably, a corresponding definition for "global" properties, the *earthmover-resilient* ones, has been defined outside the context of this thesis [24]. The combination of these two families already captures a lot of the properties of interest in the structured setting.

Surprising combinatorial phenomena Property testing in structured settings usually reveals beautiful and surprising combinatorial phenomena that were not previously evident in other contexts. The best examples in this thesis are perhaps the results in the second part, on detecting structural patterns in sequential data, and especially the non-adaptive pattern-dependent lower bounds (based on stitching and other interesting combinatorial parameters, see Chapter 3) and the structural decomposition results for monotone patterns (see Chapter 3 and, to a lesser extent, Chapter 4). This phenomenon is also clearly evident in the third part, in our study of locality, where property testing algorithms benefit from a rather unorthodox non-explicit perspective on the data via the structure of boundaries (Chapter 6) or from a new family of combinatorial arguments, so-called modification lemmas (Chapter 7).

Adaptivity Assessing to what extent adaptivity helps in general is perhaps the most important challenge in structured property testing. However, understanding adaptivity seems very difficult in general: on the one hand, non-adaptive algorithms are usually tightly connected to the combinatorial characteristics of the problem, with a smaller algorithmic component, which makes proving upper and especially lower bounds much easier. On the other hand, proving upper bounds for adaptive algorithms typically require both excellent combinatorial understanding of the problem and new algorithmic ideas, and obtaining any non-trivial lower bounds for them (beyond the most basic properties like monotonicity testing) is a major open problem. In this thesis we make a preliminary step in utilizing the power of adaptivity. Specifically, in Chapter 4, we show how strong structural characterizations can be combined with wishful thinking algorithms to yield an effective adaptive algorithm for detecting monotone patterns in sequential data.

8.1 Central Open Problems

While this thesis makes several steps forward in the systematic investigation of structured property testing, the field is still at a relatively early stage, and many central and interesting problems are still wide open. Various open problems are scattered along the different chapters, specifically in Sections 2.1, 5.1.3, 6.1.5, and 7.5. Below, we summarize the directions for future research which we believe are the most interesting.

8.1.1 The Quest for Adaptivity

Perhaps the most important challenge in this regime is to understand the power of adaptive algorithms, and how it compares to non-adaptive ones. With the possible exception of Chapter 2 (where adaptivity doesn't help much [24, 86]), it is plausible that for all main directions of research explored in this thesis, adaptivity should help immensely. Here we present several central open questions regarding adaptivity.

Adaptive detection of patterns in sequential data The first open question, posed by Newman et al. [108, 109] and discussed in detail in Chapter 5, asks the following.

> *Is it true that for any order pattern π of fixed length, detecting a π-copy in a sequence that is $\Theta(1)$-far from π-freeness requires a number of queries that is only polylogarithmic in the sequence length?*

A positive answer (which we believe holds here) would be a major breakthrough, not only because of the conceptual message that structure can be explored adaptively very effectively, but also because non-adaptive algorithms are very weak here: in Chapter 5 we have seen that most patterns of length k require $\Theta(n^{1-1/(k-\Theta(1))})$ non-adaptive queries, a minimal improvement over the trivial sampling-based algorithm.

Adaptivity for local properties The other major questions on adaptivity relate to local properties. In Chapter 6, we mention that the non-adaptive generic algorithm for local properties is optimal among non-adaptive algorithms for any fixed dimensionality d, and ask whether adaptive algorithms can break this barrier. Specifically, we ask the following.

> *Is it true that any $O(1)$-local property of d-dimensional arrays over $[n]$ is testable with $f(d) \cdot g(n)$ adaptive queries, where $f(d)$ depends only on d, and $g(n)$ depends only (reasonably) on n?*

This should be compared to the tight $\Theta(n^{d-1})$ non-adaptive query complexity, and would imply that an effective domain size reduction can be achieved with adaptivity.

Adaptivity and convexity Finally, the problem of convexity testing poses a major challenge with respect to adaptivity. This important property, very relevant for the field of optimization, seems very challenging to attack from the property testing perspective. Our non-adaptive upper bound of $O(n^{d-1})$ is the first sublinear upper bound known for convexity testing in multiple dimensions, and was very recently shown to be tight non-adaptively for $d = 2$ by Belovs, Blais and Bommireddi [20]. However, for adaptive algorithms no such limitations are known, and the following was asked in [20]:

Can two-dimensional convexity testing be conducted with a polylogarithmic number of adaptive queries?

This would, again, imply an exponential separation from the $\Theta(n)$ non-adaptive query complexity. In higher-dimensional convexity testing, one may guess (see [20]) that the adaptive query complexity is of the form $f(d) \cdot g(n)$ as suggested above, where $g(n)$ might be only polylogarithmic in n, and $f(d)$ is at most exponential (and perhaps even polynomial) in d. In contrast, for non-adaptive algorithms, an $n^{\Omega(d)}$ lower bound is proved in [20].

8.1.2 Better Structural Understanding

In several occasions throughout the thesis, the structural understanding we currently have leaves much to be desired. In these cases, better structural understanding would lead to more efficient algorithms or to a wider applicability of the currently known algorithms.

Efficient matrix removal lemma The results in Chapter 2 show that any hereditary property of ordered graphs and matrices is testable with a constant number of queries. However, this number of queries, while technically independent of the graph or matrix size, is enormous: it is at least a wowzer (tower of tower) type in the proximity parameter ε, as it relies on strong variants of Szemerédi regularity lemma. In 2007, Alon, Fischer, and Newman [8] devised an efficient regularity lemma for ordered binary matrices, where the dependence between the parameters is polynomial. They used this lemma to show that any *unordered* binary matrix property characterized as F-freeness for a finite family $\mathcal{F}$ of forbidden submatrices has a removal lemma with polynomial dependence in ε, which leads to poly$(1/\varepsilon)$-query tests for all such properties. They posed the ordered analogue as an open problem. In [6], some progress is made towards settling this question, by proving a removal lemma for all "semi-ordered" properties of this type. The problem in its full generality, of proving efficient removal lemmas for the ordered case, is however wide open. We phrase a seemingly simple special case of this question from [8], where the forbidden family $\mathcal{F}$ consists of just a single forbidden submatrix F.

Consider the property of F-freeness in binary matrices, where F is a single, fixed size forbidden matrix. Does this property satisfy a removal lemma where the dependence between the parameters is polynomial? if not, what about an exponential dependence?

We conclude by mentioning that the proof from Chapter 2 can be combined with the efficient regularity lemma from [8] to obtain a tower-type bound (instead of a wowzer-type one) for binary matrices. Perhaps the first step would be to refine this proof so as to obtain a bound with a tower of constant height (e.g., a doubly-exponential bound).

Extending the modification lemma In Chapter 7, we prove that nearly all consecutive patterns in d dimensions (except for the almost homogeneous ones), of side length at least $2 \cdot 3^d$, satisfy a modification lemma. As the only known lower bound asserts that the side length should be at least 3, there is a huge gap here. It is thus natural to ask the following.

> *Can one obtain a full characterization of those patterns that satisfy a modification lemma? In particular, is it true that for any pattern of side length at least, say, 10, which is not almost homogeneous, a modification lemma holds?*

As the property of approximate pattern freeness described in the same chapter is also of interest, we ask the following.

> *In what situations can a modification lemma type argument be proved for the property of approximate pattern freeness?*

Modification lemmas also seem relevant for applications in computational biology, where one wishes to efficiently "clean" a genetic sequence from instances of forbidden patterns, e.g. due to the need to produce sequences without problematic substrings that can trigger the immune system. We leave the general task of exploring modification lemmas in computational biology for future research.

Pattern-dependent upper bounds for sequential data The results in Chapter 5 demonstrate a pattern-dependent lower bound for detecting general (i.e., non-monotone) patterns with non-adaptive algorithms. However, at this point the generic algorithms we have are only known to be optimal for the hardest patterns, and do not take the structure of the pattern into account. This is in contrast to the lower bound side, where we have a relatively good understanding of non-adaptive algorithms, via the unique signed partition number (USPN) parameter that we devise. To summarize, our question here is as follows.

> *How can one devise non-adaptive algorithms (upper bounds) for sequential pattern detection that take the structure of the particular pattern into account?*

Bibliography

[1] P. Afshani, K. Matulef, and B. T. Wilkinson. Property testing on linked lists. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:187, 2013.

[2] N. Ailon, B. Chazelle, C. Seshadhri, and D. Liu. Estimating the distance to a monotone function. *Random Structures & Algorithms*, 31:371–383, 2007.

[3] N. Alon. Testing subgraphs in large graphs. *Random Structures & Algorithms*, 21:359–370, 2002.

[4] N. Alon, O. Ben-Eliezer, and E. Fischer. Testing hereditary properties of ordered graphs and matrices. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 848–858, 2017.

[5] N. Alon, R. A. Duke, H. Lefmann, V. Rödl, and R. Yuster. The algorithmic aspects of the regularity lemma. *Journal of Algorithms*, 16:80–109, 1994.

[6] N. Alon and O. Ben Eliezer. Efficient removal lemmas for matrices. *Order*, 37:83–101, 2020.

[7] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.

[8] N. Alon, E. Fischer, and I. Newman. Efficient testing of bipartite graphs for forbidden induced subgraphs. *SIAM Journal on Computing*, 37:959–976, 2007.

[9] N. Alon, E. Fischer, I. Newman, and A. Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. *SIAM Journal on Computing*, 39:143–167, 2009.

[10] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. *SIAM Journal on Computing*, 30:1842–1862, 2001.

[11] N. Alon and A. Shapira. Every monotone graph property is testable. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 128–137, 2005.

[12] N. Alon and A. Shapira. A characterization of the (natural) graph properties testable with one-sided error. *SIAM Journal on Computing*, 37:1703–1727, 2008.

[13] N. Alon and J.H. Spencer. *The Probabilistic Method.* Wiley Publishing, 4th edition, 2016.

[14] A. Amir and G. Benson. Two-dimensional periodicity in rectangular arrays. *SIAM Journal on Computing*, 27:90–106, 1998.

[15] A. Amir, G. Benson, and M. Farach. An alphabet independent approach to two-dimensional pattern matching. *SIAM Journal on Computing*, 23:313–323, 1994.

[16] P. Awasthi, M. Jha, M. Molinaro, and S. Raskhodnikova. Testing Lipschitz functions on hypergrid domains. *Algorithmica*, 74:1055–1081, 2012.

[17] M. Axenovich, Y. Person, and S. Puzynina. A regularity lemma and twins in words. *Journal of Combinatorial Theory, Series A*, 120:733–743, 2013.

[18] A. Belovs. Adaptive Lower Bound for Testing Monotonicity on the Line. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 31:1–31:10, 2018.

[19] A. Belovs and E. Blais. Quantum algorithm for monotonicity testing on the hypercube. *Theory of Computing*, 11:403–412, 2015.

[20] A. Belovs, E. Blais, and A. Bommireddi. Testing convexity of functions over finite domains. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2030–2045, 2020.

[21] O. Ben-Eliezer. Testing local properties of arrays. In *Proceedings of the 10th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 11:1–11:20, 2019.

[22] O. Ben-Eliezer, C. Canonne, S. Letzter, and E. Waingarten. Finding monotone patterns in sublinear time. In *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1469–1494, 2019.

[23] O. Ben-Eliezer and C. L. Canonne. Improved bounds for testing forbidden order patterns. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2093–2112, 2018.

[24] O. Ben-Eliezer and E. Fischer. Earthmover resilience and testing in ordered structures. In *Proceedings of the 33rd Computational Complexity Conference (CCC)*, pages 18:1–18:35, 2018.

[25] O. Ben-Eliezer, E. Fischer, A. Levi, and Y. Yoshida. Limits of ordered graphs and their applications. *arXiv preprint arXiv:1811-02023*, 2018.

[26] O. Ben-Eliezer, S. Korman, and D. Reichman. Deleting and testing forbidden patterns in multi-dimensional arrays. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP)*, 2017.

[27] O. Ben-Eliezer, S. Letzter, and E. Waingarten. Optimal adaptive detection of monotone patterns. *arXiv preprint arXiv:1911-01169*, 2019.

[28] B. A. Berendsohn, L. Kozma, and D. Marx. Finding and counting permutations via CSPs. In *14th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 148, pages 1:1–1:16, 2019.

[29] P. Berman, M. Murzabulatov, and S. Raskhodnikova. Constant-time testing and learning of image properties. *arXiv preprint arXiv:1503-01363*, 2015.

[30] P. Berman, M. Murzabulatov, and S. Raskhodnikova. Tolerant testers of image properties. In *Proceedings of the 43th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 90:1–90:14, 2016.

[31] P. Berman, M. Murzabulatov, and S. Raskhodnikova. Testing convexity of figures under the uniform distribution. *Random Structures & Algorithms*, 54:413–443, 2019.

[32] P. Berman, S. Raskhodnikova, and G. Yaroslavtsev. L_p-testing. In *Proceedings of the 46th ACM Symposium on the Theory of Computing (STOC)*, pages 164–173, 2014.

[33] H. Black, D. Chakrabarty, and C. Seshadhri. A $o(d) \cdot \mathrm{polylog} n$ monotonicity tester for boolean functions over the hypergrid $[n]^d$. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2133–2151, 2018.

[34] E. Blais and A. Bommireddi. Testing submodularity and other properties of valuation functions. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 33:1–33:17, 2017.

[35] E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21:311–358, 2012.

[36] E. Blais, S. Raskhodnikova, and G. Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *Proceedings of the 29th Conference on Computational Complexity (CCC)*, pages 309–320, 2014.

[37] C. Borgs, J. Chayes, L. Lovász, V. T. Sós, B. Szegedy, and K. Vesztergombi. Graph limits and parameter testing. In *Proceedings of the 38th ACM Symposium on the Theory of Computing (STOC)*, pages 261–270, 2006.

[38] R. S. Boyer and J. S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20:762–772, 1977.

[39] A. Brandstadt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey.* Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999.

[40] J. Briët, S. Chakraborty, D. García-Soriano, and A. Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32:35–53, 2012.

[41] C. L. Canonne, E. Grigorescu, S. Guo, A. Kumar, and K. Wimmer. Testing k-monotonicity. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 29:1–29:21, 2017.

[42] C. L. Canonne and Tom Gur. An adaptivity hierarchy theorem for property testing. *Computational complexity*, 27:671–716, 2018.

[43] D. Chakrabarty. Monotonicity testing. In M. Kao, editor, *Encyclopedia of Algorithms*, pages 1352–1356. Springer Berlin Heidelberg, 2014.

[44] D. Chakrabarty, K. Dixit, M. Jha, and C. Seshadhri. Property testing on product distributions: optimal testers for bounded derivative properties. *ACM Transactions on Algorithms*, 13:20:1–20:30, 2017.

[45] D. Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proceedings of the 45th ACM Symposium on the Theory of Computing (STOC)*, pages 419–428, 2013.

[46] D. Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014.

[47] D. Chakrabarty and C. Seshadhri. An $o(n)$ monotonicity tester for boolean functions over the hypercube. *SIAM Journal on Computing*, 45:461–472, 2016.

[48] D. Chakrabarty and C. Seshadhri. Adaptive boolean monotonicity testing in total influence time. In *Proceedings of the 10th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 20:1–20:7, 2019.

[49] T. M. Chan, S. Golan, T. Kociumaka, T. Kopelowitz, and E. Porat. Approximating text-to-pattern hamming distances. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 643–656, 2020.

[50] X. Chen, A. De, R. A. Servedio, and L. Tan. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC)*, pages 519–528, 2015.

[51] X. Chen, A. Freilich, R. A. Servedio, and T. Sun. Sample-based high-dimensional convexity testing. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 37:1–37:20, 2017.

[52] X. Chen, R. A. Servedio, and L. Tan. New algorithms and lower bounds for monotonicity testing. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 285–295, 2014.

[53] X. Chen, E. Waingarten, and J. Xie. Beyond Talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proceedings of the 49th ACM Symposium on the Theory of Computing (STOC)*, pages 523–536, 2017.

[54] Richard Cole. Tight bounds on the complexity of the boyer–moore string matching algorithm. *SIAM Journal on Computing*, 23:1075–1091, 1994.

[55] D. Conlon and J. Fox. Bounds for graph regularity and removal lemmas. *Geometric and Functional Analysis*, 22:1191–1256, 2012.

[56] D. Conlon and J. Fox. Graph removal lemmas. In *Surveys in Combinatorics*, 2013.

[57] D. Conlon, J. Fox, C. Lee, and B.Sudakov. Ordered ramsey numbers. *Journal of Combinatorial Theory, Series B*, 122:353–383, 2017.

[58] M. Crochemore, A. Czumaj, L. Gasieniec, S. Jarominek, T. Lecroq, W. Plandowski, and W. Rytter. Speeding up two string-matching algorithms. *Algorithmica*, 12:247–267, 1994.

[59] P. Damaschke. Forbidden ordered subgraphs. In Rainer Bodendiek and Rudolf Henn, editors, *Topics in Combinatorics and Graph Theory: Essays in Honour of Gerhard Ringel*, pages 219–229. Physica-Verlag HD, 1990.

[60] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–166, 1950.

[61] P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

[62] F. Ergün and H. Jowhari. On the monotonicity of a data stream. *Combinatorica*, 35:641–653, 2015.

[63] F. Ergün, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Vishwanthan. Spot-checkers. *Journal of Computer and System Sciences*, 60:717–751, 2000.

[64] C. Even-Zohar and C. Leng. Counting small permutation patterns. *arXiv preprint arXiv:1911-01414*, 2019.

[65] U. Feige, T. Koren, and M. Tennenholtz. Chasing ghosts: Competing with stateful policies. *SIAM Journal on Computing*, 46:190–223, 2017.

[66] E. Fischer. On the strength of comparisons in property testing. *Information and Computation*, 189:107–116, 2004.

[67] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC)*, pages 474–483, 2002.

[68] E. Fischer and I. Newman. Testing of matrix properties. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing (STOC)*, pages 286–295, 2001.

[69] E. Fischer and I. Newman. Testing of matrix-poset properties. *Combinatorica*, 27:293–327, 2007.

[70] E. Fischer and I. Newman. Testing versus estimation of graph properties. *SIAM Journal on Computing*, 37:482–501, 2007.

[71] E. Fischer and E. Rozenberg. Lower bounds for testing forbidden induced substructures in bipartite-graph-like combinatorial objects. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 464–478, 2007.

[72] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12:133–137, 1981.

[73] J. Fox. A new proof of the graph removal lemma. *Annals of Mathematics*, 174:561–579, 2011.

[74] J. Fox. Stanley–Wilf limits are typically exponential. *arXiv preprint arXiv:1310-8378*, 2013. Also: Advances in Mathematics, to appear.

[75] J. Fox and L. M. Lovász. A tight lower bound for szemerédi's regularity lemma. *Combinatorica*, 37:911–951, 2017.

[76] A. Gál and P. Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. *SIAM Journal on Computing*, 39:3463–3479, 2010.

[77] Z. Galil, J. G. Park, and K. Park. Three-dimensional periodicity and its application to pattern matching. *SIAM Journal of Discrete Mathematics*, 18:362–381, 2005.

[78] Z. Galil and J. Seiferas. Time-space-optimal string matching. *Journal of Computer and System Sciences*, 26:280–294, 1983.

[79] L. Gishboliner and A. Shapira. Removal lemmas with polynomial bounds. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 510–522, 2017.

[80] O. Goldreich, editor. *Property Testing - Current Research and Surveys [outgrow of a workshop at the Institute for Computer Science (ITCS) at Tsinghua University, January 2010]*, volume 6390 of *Lecture Notes in Computer Science*. Springer, 2010.

[81] O. Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.

[82] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samordinsky. Testing monotonicity. *Combinatorica*, 20:301–337, 2000.

[83] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998.

[84] O. Goldreich and Tali Kaufman. Proximity oblivious testing and the role of invariances. In O. Goldreich, editor, *Studies in Complexity and Cryptography*, pages 173–190. Springer Berlin Heidelberg, 2011.

[85] O. Goldreich and D. Ron. On proximity-oblivious testing. *SIAM Journal on Computing*, 40:534–566, 2011.

[86] O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. *Random Structures & Algorithms*, 23:23–57, 2003.

[87] P. Gopalan, T. S. Jayram, R. Krauthgamer, and R. Kumar. Estimating the sortedness of a data stream. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 318–327, 2007.

[88] W.T. Gowers. Lower bounds of tower type for szemerédi's uniformity lemma. *Geometric and Functional Analysis*, 7:322–337, 1997.

[89] E. Grigorescu, A. Kumar, and K. Wimmer. Flipping out with Many Flips: Hardness of Testing k-Monotonicity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 40:1–40:17, 2018.

[90] S. Guillemot and D. Marx. Finding small patterns in permutations in linear time. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 82–101, 2014.

[91] G. Higman. Ordering by Divisibility in Abstract Algebras. *Proceedings of the London Mathematical Society*, s3-2:326–336, 1952.

[92] M. Jha and S. Raskhodnikova. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM Journal on Computing*, 42:700–731, 2013.

[93] S. Kalyanasundaram and A. Shapira. A wowzer type lower bound for the strong regularity lemma. *Proceedings of the London Mathematical Society*, 106:621–649, 07 2011.

[94] J. Kärkkäinen and E. Ukkonen. Multidimensional string matching. In M. Kao, editor, *Encyclopedia of Algorithms*, pages 559–562. Springer US, 2008.

[95] S. Khot, D. Minzer, and M. Safra. On monotonicity testing and boolean isoperimetric type theorems. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 52–58, 2015.

[96] I. Kleiner, D. Keren, I. Newman, and O. Ben-Zwi. Applying property testing to an image partitioning problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:256–265, 2011.

[97] D. E. Knuth. *The Art of Computer Programming: Volume I: Fundamental Algorithms.* Pearson Education, 1997.

[98] D. E. Knuth, J. H. Morris, Jr., and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6:323–350, 1977.

[99] S. Korman, D. Reichman, G. Tsur, and S. Avidan. Fast-match: Fast affine template matching. *International Journal of Computer Vision*, 121:111–125, 2017.

[100] T. Lecroq. Fast exact string matching algorithms. *Information Processing Letters*, 102:229–235, 2007.

[101] L. Lovász. *Large Networks and Graph Limits.* American Mathematical Society colloquium publications. American Mathematical Society, 2012.

[102] L. Lovász and B. Szegedy. Szemerédi's lemma for the analyst. *Geometric and Functional Analysis*, 17:252–270, 2007.

[103] S. Moriguchi and K. Murota. On discrete hessian matrix and convex extensibility. *Journal of the Operations Research Society of Japan*, 1:48–62, 2012.

[104] G. Moshkovitz and A. Shapira. A short proof of Gowers' lower bound for the regularity lemma. *Combinatorica*, 36:187–194, 2016.

[105] K. Murota. *Discrete Convex Analysis.* Society for Industrial and Applied Mathematics, 2003.

[106] B. Nagle, V. Rödl, and M. Schacht. The counting lemma for regular k-uniform hypergraphs. *Random Structures & Algorithms*, 28:113–179, 2006.

[107] T. Naumovitz and M. E. Saks. A polylogarithmic space deterministic streaming algorithm for approximating distance to monotonicity. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1252–1262, 2015.

[108] I. Newman, Y. Rabinovich, D. Rajendraprasad, and C. Sohler. Testing for forbidden order patterns in an array. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1582–1597, 2017.

[109] I. Newman, Y. Rabinovich, D. Rajendraprasad, and C. Sohler. Testing for forbidden order patterns in an array. *Random Structures & Algorithms*, 55(2):402–426, 2019. Preliminary version in SODA'17 [108].

[110] I. Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM Journal on Computing*, 42:1095–1112, 2013.

[111] R. K. S. Pallavoor, S. Raskhodnikova, and N. M. Varma. Parameterized property testing of functions. *ACM Transactions on Computation Theory*, 9:17:1–17:19, 2018.

[112] M. Parnas, D. Ron, and R. Rubinfeld. On testing convexity and submodularity. *SIAM Journal on Computing*, 32:1158–1184, 2003.

[113] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72:1012–1042, 2006.

[114] L. Rademacher and S. Vempala. Testing geometric convexity. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 469–480, 2004.

[115] S. Raskhodnikova. Approximate testing of visual properties. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 370–381, 2003.

[116] S. Raskhodnikova. Testing if an array is sorted. In M. Kao, editor, *Encyclopedia of Algorithms*, pages 2219–2222. Springer Berlin Heidelberg, 2014.

[117] R. L. Rivest. On the worst-case behavior of string-searching algorithms. *SIAM Journal on Computing*, 6:669–674, 1977.

[118] V. Rödl and J. Skokan. Regularity lemma for k-uniform hypergraphs. *Random Structures & Algorithms*, 25:1–42, 2004.

[119] D. Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, 1:307–402, 2008.

[120] D. Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5:73–205, 2009.

[121] D. Ron and G. Tsur. Testing properties of sparse images. *ACM Transactions on Algorithms*, 10, 2014.

[122] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25:252–271, 1996.

[123] A. Rubinstein, S. Seddighin, Z. Song, and X. Sun. Approximation algorithms for lcs and lis with truly improved running times. In *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1121–1145, 2019.

[124] I. Z. Ruzsa and E. Szemerédi. Triple systems with no six points carrying three triangles. *Combinatorics (Proceedings of the Fifth Hungarian Colloquium, Keszthely, 1976)*, 2:939–945, 1978.

[125] M. Saks and C. Seshadhri. Space efficient streaming algorithms for the distance to monotonicity and asymmetric edit distance. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1698–1709, 2013.

[126] M. Saks and C. Seshadhri. Estimating the longest increasing sequence in polylogarithmic time. *SIAM J. Comput.*, 46:774–823, 2017.

[127] C. Seshadhri and J. Vondrák. Is submodularity testable? *Algorithmica*, 69:1–25, 2010.

[128] R. Simion and F. W. Schmidt. Restricted permutations. *European Journal of Combinatorics*, 6:383–406, 1985.

[129] M. Sudan. Invariance in property testing. In O. Goldreich, editor, *Property Testing: Current Research and Surveys*, pages 211–227. Springer Berlin Heidelberg, 2010.

[130] X. Sun and D. P. Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 336–345, 2007.

[131] E. Szemerédi. Regular partitions of graphs. *Problèmes combinatoires et théorie des graphes (Interbational Coloquium of the CNRS, Orsay, 1976)*, pages 399–401, 1978.

[132] T. Tao. A variant of the hypergraph removal lemma. *Journal of Combinatorial Theory, Series A*, 113:1257–1280, 2006.

[133] T. Tao. *Structure and Randomness: Pages from Year One of a Mathematical Blog.* American Mathematical Society, 2008.

[134] P. Uznański. Approximating Text-To-Pattern Distance via Dimensionality Reduction. In *31st Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 29:1–29:11, 2020.

[135] P. Valiant. Testing symmetric properties of distributions. In *Proceedings of the 40th ACM Symposium on the Theory of Computing (STOC)*, pages 383–392, 2008.

[136] J. van Leeuwen. Graph algorithms. In *Handbook of Theoretical Computer Science (Vol. A): Algorithms and Complexity*, pages 525–631. MIT Press, 1991.

[137] V. Vatter. Permutation classes. In *Handbook of Enumerative Combinatorics*, chapter 12. CRC Press, 2015.

[138] Y.Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 97–108, 1999.

תכונה זו היא מקומית, עם משפחה אסורה בעלת איבר אחד בלבד. המוטיבציה לחקר תכונה זו מגיעה מתחומים כגון ביולוגיה חישובית או ראיה ממוחשבת, בהם יש צורך משמעותי באלגוריתמים יעילים למציאת תבניות מקומיות בכמות גדולה של מידע. התוצאה המרכזית כאן היא אלגוריתם בדיקה יעיל לתכונה של אי הכלת תבנית P. מספר הדגימות של האלגוריתם תלוי במימד d וכן לינארית ב-$1/\varepsilon$ אך אינו תלוי ב-n. בנוסף אנו מציגים אלגוריתם לחישוב מדויק של המרחק מקיום התכונה בחד-מימד; במימדים גבוהים יותר, בעיה זו היא NP-קשה [72]. הבעיה של חישוב מדויק של המרחק ביעילות הינה רלוונטית בביולוגיה חישובית, לצרכים של ניקוי סדרות מידע גנטי מתבניות בעייתיות תוך מספר קטן של שינויים ככל האפשר.

תשתית מתמטית. הגרעין המתמטי של ההוכחה, שאנו מכנים *למת השינוי*, הוא הטענה הבאה: כמעט לכל תבנית גדולה מספיק P, ולכל מערך A המכיל מופע של P כתבנית רצופה, קיים ערך במיקום כלשהו במופע זה של P, ששינויו לא יגרום ליצירת עותקים חדשים של P. מלמת שינוי כזו ניתן להוכיח למת הסרה, שכן כמסקנה ישירה שלה, מספר מופעי P ב-A הוא לפחות כמרחקה A מאי הכלה של P כתבנית רצופה.

מקורות: התוצאות המוצגות בחלק זה מבוססות על המאמר הבא.

O. Ben-Eliezer, S. Korman, D.Reichman, *Deleting and testing forbidden patterns in multi-dimensional arrays,* Proc. 44th International Colloquium on Automata, Languages and Programming (*ICALP*), 2017, 9:1--9:14.

אלגוריתם כללי לתכונות מקומיות

התוצאות המתוארות בפסקה זו מופיעות בפרק 6.

התוצאה המרכזית בפרק 6 היא אלגוריתם כללי, עם שגיאה חד-צדדית, לבדיקת כל התכונות ה-k-מקומיות ב-d מימדים, מעל כל אלפבית Σ סופי (לא בהכרח בגודל קבוע). מספר הדגימות הנדרש הוא $O\left(\frac{k}{\varepsilon}\cdot\log\frac{\varepsilon n}{k}\right)$ במקרה החד-מימדי, ו-$\frac{k}{\varepsilon^{1/d}}\cdot\left(O(n)\right)^{d-1}$ עבור $d>1$ מימדים. בפרט, עבור k,ε קבועים, החסמים המתאימים הינם $O(\log n)$ ו- $O(n^{d-1})$ בהתאמה. החסמים המתקבלים הדוקים עבור מגוון רחב של תכונות, במימד אחד וכן במימדים גבוהים יותר. כמקרה פרטי של אלגוריתם כללי זה, מתקבלים אלגוריתמים תת-לינאריים לבדיקת קמירות ותת-מודולוריות במימדים גבוהים. אלו האלגוריתמים התת-לינאריים הראשונים לבדיקת תכונות אלו [20].

תשתית מתמטית. נתבונן בהוכחה הקלאסית של ארגון ואחרים [63] לכך שמונוטוניות ניתנת לבדיקה באמצעות $O(\log n)$ דגימות לא אדפטיביות. נבחר איבר אקראי $i\in[n]$, ונבצע חיפוש בינארי דמיוני על איברי $[n]$, שנקודת ההתחלה שלו היא מרכז המערך והוא מסתיים ב-i. תהא $Q\subseteq[n]$ קבוצת האיברים שבהם ביקרנו במהלך התהליך. הטענה המרכזית של [63] היא שאם הסדרה הינה ε-רחוקה ממונוטוניות, אזי דגימת כל איברי Q תמצא בהסתברות ε לפחות הוכחה לכך שהסדרה אינה מונוטונית, ובכך תגרום לדחייתה על ידי האלגוריתם.

אנו מראים כי באופן מפתיע, הרעיון המתואר לעיל לבדיקת מונוטוניות מתאים לבדיקת כל תכונה מקומית. לשם נוחות, נתבונן במקרה הפשוט ביותר, בו $d=1,k=2$. נאמר שתת-מערך עוקב *אינו בר תיקון* אם, כאשר מקבעים את ערכי הקצה של תת המערך (ערך ההתחלה וערך הסיום), אך מאפשרים לשנות את ערכי הפנים שלו, כל בחירה של ערכי פנים תייצר תת מערך *שאינו* מקיים את התכונה. במונוטוניות, ההגדרה הזו טבעית: תת-מערך אינו בר תיקון אם האיבר הראשון בו גדול בערכו מהאיבר האחרון. ההוכחה המתוארת עבור בדיקת מונוטוניות תתאים לכל תכונה מונוטונית, כאשר במקום לבדוק האם אוסף האיברים Q הינו מונוטוני, מוודאים כי כל תתי-האינטרוולים בו ברי תיקון.

מקורות: התוצאות המוצגות בחלק זה מבוססות על המאמר הבא.

O. Ben-Eliezer, *Testing local properties of arrays*, Proc. 10th Innovations in Theoretical Computer Science (*ITCS*), 2019, 11:1--11:20.

למת השינוי ובדיקת אי-הכלת תבנית רצופה

התוצאות המתוארות בפסקה זו מופיעות בפרק 7.

האלגוריתם הכללי עבור תכונות מקומיות מתאים לטווח רחב מאוד של תכונות, ובאופן טבעי עבור רבות מהתכונות הללו, הוא אינו אופטימלי. כעת נתמקד בתכונה אחת כזו – התכונה של אי הכלת תבנית רצופה ספציפית P.

מההיבט הקומבינטורי, החסמים התחתונים שאנו מוכיחים כאן מציגים פרמטר חדש של פרמוטציות, *מספר החלוקה המסומנת הייחודית* (unique signed partition number), שטרם הוגדר או נחקר בעבר. בעוד שפרמטר זה מסובך למדי לתיאור, וריאציה פשוטה יותר שלו, *מספר התפירה* (stitching number) שנסמנו כאן $s(\pi)$ נותן תוצאות מעט חלשות יותר אך קל יותר להסבר, ונציג אותו בקצרה כאן. עבור פרמוטציה בה (בלי הגבלת הכלליות) הערך המינימלי 1 מופיע לפני הערך המקסימלי k, מספר התפירה הוא גודל האוסף המינימלי של זוגות איברים עוקבים בפרמוטציה, שערכיהם עולים, כך שלכל איבר $i \in [k]$ קיים זוג עוקב כנ"ל עבורו $a \leq i \leq b$. ניתן לראות כי מספר התפירה של כמעט כל התבניות באורך k הינו 2 או 3. מצד שני, מקרה פרטי של החסם התחתון הלא-אדפטיבי שלנו מראה כי לכל תבנית π, מספר הדגימות הנדרש הינו $\Omega\left(n^{1-\frac{1}{k-s(\pi)}}\right)$. בחיבור שתי תוצאות אלו, מתקבל החסם התחתון של $\Omega\left(n^{1-\frac{1}{k-3}}\right)$ עבור כמעט כל התבניות מאורך k.

מקורות: התוצאות המוצגות בחלק זה מבוססות על המאמר הבא.

O. Ben-Eliezer, C. Canonne, *Improved bounds for testing forbidden order patterns, Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2018, 2093--2112.*

חלק שלישי: בדיקת תכונות מקומיות במבנים סדורים

בחלק האחרון של התזה (פרקים 6-7), נעסוק בבדיקת תכונות מקומיות במערכים חד-מימדיים ורב-מימדיים. מערך d-מימדי מעל אלפבית Σ הינו פונקציה מהצורה $A: [n]^d \to \Sigma$. תכונה נחשבת k-מקומית אם ניתן לאפיין אותה באופן מלא על ידי משפחה של תתי-מערכים עוקבים (או רצופים) אסורים באורך k. לדוגמה, מונוטוניות בחד-מימד (ולמעשה, גם במימדים גבוהים יותר) היא תכונה 2-מקומית, כיוון שסדרה היא מונוטונית לא יורדת אם ורק אם היא אינה מכילה תת מערך עוקב מהצורה (a, b) עבור $a > b$.

רבות מהתכונות הנחקרות ביותר בספרות של תחום בדיקת התכונות הן מקומיות. כך למשל, מונוטוניות ורציפות ליפשיץ הן 2-מקומיות; קמירות דיסקרטית היא 3- או 4-מקומית בדרך כלל, כתלות בהגדרה; באופן כללי, תכונות המוגדרות על ידי הנגזרת הדיסקרטית ה-k הן $(k+1)$-מקומיות. תת-מודולריות היא 2-מקומית; ותכונות רבות בתחומים אפליקטיביים יותר, כמו ביולוגיה חישובית וראייה ממוחשבת, הן k-מקומיות עבור k קטן.

בחלק זה, נראה כי כל התכונות המקומיות ניתנות לבדיקה במספר דגימות תת-לינארי. בנוסף, באמצעות ניתוח קומבינטורי מעמיק יותר של המקרה בו המשפחה האסורה (בהגדרת המקומיות) מכילה תת-מערך אחד בלבד, חסמים משופרים מושגים עבור התכונה של "אי הכלת תת מערך עוקב ספציפי".

במקרה הכאוטי, כמות הפרופילים השונים גדולה, וגורמת בהכרח ליצירת מספר גדול של סדרות מונוטוניות ארוכות וקלות למציאה באמצעות $O(\log n)$ דגימות, אפילו ללא אדפטיביות.

עבור המקרה האדפטיבי, לא ניתן להפעיל ביעילות שיקולים דומים לאלו הלא-אדפטיביים, בשל התשלום הכפלי של $O(\log n)$ עבור כל עומק של הרקורסיה. במקום זאת, האלגוריתם מבוסס על גישת wishful thinking רקורסיבית, הדוגמת מספר קטן של איברים ובוחרת מתוכם זוג איברים המועמדים לתפקד כערכי הקצה (1 ו-k) של מופע אפשרי של $(1,2,\ldots,k)$, וממשיכה בחיפוש מופע באורך $k-2$ בתת הסדרה שבין איברים אלו.

מקורות: התוצאות המוצגות בחלק זה מבוססות על המאמרים הבאים.

- O. Ben-Eliezer, C. Canonne, S. Letzter, E. Waingarten, *Finding monotone patterns in sublinear time*, Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2019, 1457—1482.

- O. Ben-Eliezer, S. Letzter, E. Waingarten, *Optimal adaptive detection of monotone patterns.*

תבניות לא מונוטוניות

התוצאות המתוארות בפסקה זו מופיעות בפרק 5.

התוצאות של נוימן ושותפיו [109] מראות כי קיימות תבניות שעבורן אלגוריתמים לא אדפטיביים הם כמעט חסרי תועלת. בפרט, בעוד שכל תבנית π ניתנת לבדיקה על ידי $O\left(n^{1-\frac{1}{k}}\right)$ דגימות מקריות מהסדרה, קיימות תבניות שבמקרה הלא אדפטיבי דורשות לפחות $\Omega\left(n^{1-\frac{2}{k+1}}\right)$ דגימות. בתזה זו נשפר משמעותית את ההבנה הכללית של המקרה הלא-אדפטיבי. ראשית, נראה אלגוריתם המשפר במעט את החסם המתקבל מדגימות מקריות, ל-$O\left(n^{1-\frac{1}{k-1}}\right)$. אלגוריתם זה משלב, לצד דגימות מקריות, גם דגימות של אינטרוולים רצופים מלאים.

מצד שני, לכל k קבוע נראה כי קיימות תבניות הדורשות $\Omega\left(n^{1-\frac{1}{k-1}}\right)$ דגימות לא אדפטיביות. חסם זה הדוק. למעשה, החסם התחתון שאנו מוכיחים הינו כללי בהרבה, ומראה למשל שכמעט לכל התבניות באורך k, מספר הדגימות הלא-אדפטיביות הנדרש הינו $\Omega\left(n^{1-\frac{1}{k-3}}\right)$. כלומר, כמעט לכל התבניות באורך k, אפקטיבית, לא קיימים אלגוריתמים לא-אדפטיביים טובים משמעותית מדגימה אקראית! בנוסף, נראה כי לכל $1 \le \ell \le k-1$, קיימת תבנית באורך k שדורשת $\widetilde{\Theta}\left(n^{1-\frac{1}{\ell}}\right)$ דגימות, ובכך נענה בחיוב על החלק השני של השאלה הפתוחה של נוימן ושותפיו הנוגעת למקרה הלא-אדפטיבי.

$\pi = (k, k-1, \dots, 1)$ לבין שאר התבניות: בעוד שתבניות מונוטוניות ניתנות לבדיקה על ידי אלגוריתם לא אדפטיבי עם מספר פולילוגריתמי של דגימות, כל התבניות שאינן מונוטוניות דורשות לפחות $\Omega(\sqrt{n})$ דגימות כאשר האלגוריתם אינו אדפטיבי. מספר זה כמעט הדוק עבור $\pi = (1,3,2)$. בנוסף, עבור תבנית זו, יש אלגוריתם אדפטיבי שמספר הדגימות שלו פולילוגריתמי בלבד – כלומר, שיפור אקספוננציאלי לעומת המקרה הלא-אדפטיבי.

באותו המאמר מועלות שתי שאלות פתוחות.

1. המקרה האדפטיבי: האם כל תבנית π ניתנת לבדיקה באמצעות אלגוריתם אדפטיבי עם מספר פולילוגריתמי בלבד של תכונות?
2. המקרה הלא-אדפטיבי: כיצד משפיע מבנה התבנית π על מספר הדגימות הלא-אדפטיביות הנדרשות למציאת מופע של π? האם ישנו מספר אינסופי של תבניות π לא מונוטוניות, שעבורן מספר זה קטן מ-$O(n^{0.99})$?

בחלק זה, נציג מספר תוצאות, הן במקרה האדפטיבי והן במקרה הלא אדפטיבי. השאלה הפתוחה הראשונה, הנוגעת לאדפטיביות במקרה הכללי, היא כנראה קשה מאוד. את השאלה הפתוחה השניה אנו פותרים באופן חלקי (את חלקה השני אנו פותרים במלואו).

תבניות מונוטוניות

התוצאות המתוארות בפסקה זו מופיעות בפרקים 3-4.

התוצאות של נוימן ושותפיו [109] מראות שלתבנית המונוטונית $\pi = (1,2,\dots,k)$, קיים אלגוריתם לא אדפטיבי בעל מספר דגימות $(\log n)^{O(k^2)}$. נשאלת השאלה: מהי התלות הפולילוגריתמית הנכונה בבעיה זו? כפי שנראה, התשובה הינה $\Theta(\log n)$ למקרה האדפטיבי ו-$\Theta\left((\log n)^{\lfloor \log_2 k \rfloor}\right)$ למקרה הלא-אדפטיבי. האלגוריתם של נוימן ושותפיו מתבסס על האבחנה הבאה בנוגע לפריקות של תבניות מונוטוניות: כדי למצוא תבנית כזו באורך k, מספיק למצוא מופעים שתי תבניות קצרות יותר, באורכים $\ell, k-\ell$, שניתנות לשרשור – במובן שנקודת ההתחלה של אחד המופעים היא בעלת ערך גבוה מנקודת הסיום של המופע האחר, וכן מיקומה של נקודת התחלה זו הוא אחרי נקודת הסיום של המופע האחר. באמצעות תכונת פריקות זו, ניתן למצוא תבניות מונוטוניות באופן רקורסיבי על ידי מניה על "רוחבים" מתאימים.

כדי לשפר את החסם הנ"ל במקרה הלא-אדפטיבי, אנו מוכיחים אפיון מבני מסוג "מבנה או כאוס" (ראו למשל את הספר של טאו [133]) עבור סדרות שהינן רחוקות מאי-הכלת התבנית $(1,2,\dots,k)$. במקרה המבני, כמות גדולה ממופעי התבנית המונוטונית הם בעלי אותו "פרופיל מרחקים" טיפוסי; הפעלה איטרטיבית של המקרה המבני יוצרת "פרופיל עץ" טיפוסי, בעל k קדקדים, למרחקים האופייניים בין הערכים של מופעי $(1,2,\dots,k)$ בגרף. האבחנה האלגוריתמית המרכזית היא, שבאמצעות תשלום כפלי של $O(\log n)$ דגימות, ניתן לשבור את פרופיל העץ לעצים בגודל עד $k/2$; וכן, שבבסיס האינדוקציה, עבור עצים בגודל 1, נדרש מספר קבוע של דגימות.

מצליחה "לייצג" אובייקטים סדורים בצורה ראויה. התרומה הטכנית המשמעותית בחלק זה של התזה היא בניית תשתית רגולריות התומכת באובייקטים סדורים. מעשית, דבר זה מבוצע על ידי שזירת רגולרית לגרפים עם רגולריות למחרוזות באופן זהיר. בנוסף לבניה זו, אנו נדרשים להוכיח ולהשתמש במשפט מסגנון רמזי עבור גרפים רב-חלקיים עם צלעות "בלתי רצויות". אנו מראים כי בכל גרף גדול מספיק כנ"ל, קיים תת-גרף רב-חלקי מושרה, עם מספר נתון של קדקדים בכל חלק, כך שתת-הגרף המושרה בין כל זוג חלקים הוא מונוכרומטי, ובנוסף המספר הכולל של קשתות בלתי רצויות בתת המבנה לא גדול בהרבה מבגרף המקורי.

מקורות: התוצאות המוצגות בחלק זה מבוססות על המאמר הבא.

N. Alon, O. Ben-Eliezer, E. Fischer, *Testing hereditary properties of ordered graphs and matrices,* Proc. 58th Annual IEEE Symposium on Foundations of Computer Science (*FOCS*), 2017, 848--858.

חלק שני: אלגוריתמים תת-לינאריים למציאת תבניות במידע סדרתי

בחלק השני של התזה (פרקים 3-5) נעסוק בבעיית בדיקת תכונות במידע סדרתי (פונקציות $f: [n] \to \mathbb{R}$) המוגדרת כדלקמן. נקבע פרמוטציה $\pi: [k] \to [k]$, כאשר k קבוע. סדרה f כנ"ל מכילה מופע של π אם קיימים $1 \le i_1 < i_2 < \cdots < i_k \le n$ עבורם $f(i_j) < f(i_\ell)$ אם ורק אם $\pi(j) < \pi(\ell)$. נוימן, רבינוביץ', רג'נדרפרסד וזולר [108, 109] חקרו את הבעיה של מציאת מופע של π במידע סדרתי מנקודת המבט של בדיקת תכונות. ספציפית, הבעיה הנחקרת היא של התכונה "אי הכלת מופע של π". לצורך הדיון, נניח כי k וכן פרמטר המרחק ε שניהם קבועים, ונתמקד באלגוריתמי בדיקת תכונות עם שגיאה חד-צדדית.

המוטיבציה הטבעית ביותר לחקר הבעיה הנ"ל מגיעה מאנליזה של סדרות עתיות (time series analysis), תחום שבו המשימה המרכזית היא מציאת תבניות גלובליות במידע סדרתי שבמקרים רבים הוא עצום בגודלו. בנוסף, לבעיה זו קשרים הדוקים עם בעיות קלאסיות אחרות בסדרות, כמו בעיית תת הסדרה העולה הארוכה ביותר. המקרה הפשוט ביותר של הבעיה, בו $\pi = (2,1)$, שקול לחלוטין לבדיקת מונוטוניות – שהיא כנראה הבעיה הנחקרת ביותר בתחום בדיקת התכונות המדגמית כולו (ראו פרק 3.1).

במאמרם [109], נוימן ושותפיו מוכיחים מספר תוצאות מפתיעות בנוגע לבדיקת "אי הכלת π", המרמזות כי בבעיה זו תלות קומבינטורית מעניינת בזהותה של התבנית π, וכן תלות משמעותית במידת האדפטיביות של האלגוריתם. בהקשר המבני, התוצאות המרכזיות הינן הפרדה בין התבניות המונוטוניות $\pi = (1,2,\dots,k)$ וכן

טבעי q, כך שלכל גרף שהינו ε-רחוק מאי הכלת $\mathcal{F}$, בהסתברות טובה, תת-הגרף המושרה על q קדקדים אקראיים בגרף יכיל תת-גרף אסור מתוך $\mathcal{F}$. ההוכחה נובעת מגרסה חזקה של למת הרגולריות של סמרדי [131], הנקראת *למת הרגולריות החזקה*.
שימוש בלמות הסרה, שבתורן מוכחות בעזרת למות רגולריות, הינה השיטה המקובלת ביותר להוכחת תוצאות כלליות בבדיקת תכונות של גרפים במודל הצפוף. יש לכך סיבה טובה: תוצאות מאוחרות יותר [9] הראו כי מידת רגולריות של תכונה שקולה, במובן חזק יחסית, ליכולת לבדוק אותה באמצעות מספר קבוע של דגימות. אלון ושפירא הכלילו את למת ההסרה המושרית למשפחות אינסופיות [12]. כיוון שכל תכונה תורשתית בגרפים (כלומר, תכונה סגורה תחת הסרה של קדקדים) ניתנת לאפיון על ידי משפחה סופית או אינסופית של תתי גרפים מושרים אסורים, אנו מסיקים את התוצאה הכללית הבאה באשר לבדיקת תכונות בגרפים.

משפט ([12]): *כל תכונה תורשתית בגרפים לא מסודרים הינה בדיקה במספר קבוע של דגימות.*

למת הסרה יעילה עבור מטריצות בינאריות ללא חשיבות לסדר השורות והעמודות הוכחה על ידי אלון, פישר ונוימן [8] בשנת 2007. בהקשר זה, אנו מתייחסים לחיתוך של s שורות ו-t עמודות מתוך מטריצה (לאו דווקא רצופות) בתור תת-מטריצה מסדר $s \times t$ של המטריצה המקורית, כאשר הסדר בתת-המטריצה מושרה מזה של המטריצה המקורית. הכלי המרכזי ב[8] הינו למת רגולריות מותנית יעילה עבור מטריצות בינאריות. כיוון שלמת רגולריות זו סדורה מטבעה, באותו המאמר, הועלתה ההשערה שניתן להוכיח באמצעותה למת הסרה סדורה למטריצות שבהן סדר השורות והעמודות חשוב (בדומה למצב בגרפים, מקרה זה הוא פחות סימטרי, וההוכחה למקרה הסימטרי מתוך [8] לא תתאים עבורו).

התוצאה המרכזית בחלק הראשון של התזה היא הוכחת גרסה מוכללת של ההשערה הנזכרת לעיל. למעשה, ההוכחה תקפה גם לגרפים סדורים בעלי יותר משני צבעי קשתות אפשריים, ולמטריצות סדורות מעל כל אלפבית בגודל קבוע. בשונה מהתוצאות הקודמות שתוארו כאן, הוכחת טענה זו אינה דורשת סימטריה כלל.

משפט: *כל תכונה תורשתית של גרפים או מטריצות מעל אלפבית קבוע מקיימת למת הסרה, ולכן גם ניתנת לבדיקה במספר קבוע של דגימות. טענה זו נכונה לתכונות סדורות ולא-סדורות כאחד.*

תשתית מתמטית. הוכחת למות הסרה בדרך כלל דורשת בניית "סכמת רגולריות" מתאימה: מבנה בגודל קבוע המקיים שתי תכונות מרכזיות. הראשונה היא דמיון משמעותי לגרף המקורי, במובן זה שניתן לשנות את הגרף המקורי לניפוח של הסכמה באמצעות הוספת והסרת מספר קטן מאוד של קשתות. השניה היא ייצוגיות, במובן כזה שכל תופעה רלוונטית שניתן לראות בגרף הסכמטי, קיימת גם בגרף המקורי.

למת ההסרה הבסיסית ביותר משתמשת בלמת הרגולריות של סמרדי בתור סכמת רגולריות [131]. במקרה המסובך יותר של למת ההסרה המושרית, יש צורך בסכמה מסובכת יותר. יחד עם זאת, אפילו סכמה זו אינה

בתזה זו, אנו מנסים לבסס הבנה שיטתית כיצד לייצר אלגוריתמי בדיקת תכונות אפקטיביים עבור מידע סדור, לא סימטרי. בתזה נדון במספר תוצאות רלוונטיות, החל בשבירת "מחסום הסימטריה" עבור בדיקת תכונות בגרפים, דרך מציאת אלגוריתמים והוכחת חסמים תחתונים אופטימיליים לבעיות מציאת תבניות במידע סדרתי, וכלה בגילוי כלים בסיסיים חדשים לפתרון בעיות מקומיות במידע סדור. לכל הבעיות הנדונות בתזה זו מבנה קומבינטורי יפה ומעניין, שהבנתו חיונית לפתרון הבעיה. חלק מהשיטות המוצגות כאן גרמו לפיתוחם של מינוחים קומבינטוריים חדשים, מעניינים לכשלעצמם.

לתזה זו שבעה פרקים המחולקים לשלושה חלקים. בחלק הראשון, נדון בבדיקת תכונת בגרפים מסודרים (ללא סימטריה בין הקדקדים). החלק השני דן בבדיקת תכונות מבניות של מידע סדרתי. החלק השלישי עוסק בבדיקת תכונות סדורות מקומיות. לכל אחד מהחלקים, נציג כעת את הרקע הכללי לבעיה, הפתרון האלגוריתמי, ונקודת המבט הקומבינטורית.
מכאן ואילך נניח כי הקורא\ת מכיר\ה את המינוחים וההגדרות הסטנדרטיים בתחום בדיקת התכונות. מידע נוסף מופיע בפרק 1.4.

חלק ראשון: בדיקת תכונות בגרפים סדורים

התוצאות המתוארות בפסקה זו מופיעות בפרק 2.

בחלק הראשון של התזה, נפתח למות הסרה (שבתורן מובילות לאלגוריתמים יעילים) עבור מגוון רחב של תכונות בגרפים ומטריצות, ללא צורך בסימטריה. מספר ניכר של עבודות בתחום בדיקת התכונות עסק באפיון התכונות הבדיקות ביעילות (בדרך כלל במספר קבוע של דגימות) עבור גרפים לא סדורים, שבהם ישנה סימטריה בין הקדקדים. עם זאת, עבודות אלו לא טיפלו במקרה בו לקדקדים יש סדר מובנה ובלתי ניתן לשינוי, למשל בגרפים סדורים או בתמונות (שניתן לייצגן כגרפים דו צדדים סדורים בעלי מגוון צבעי קשתות, כאשר הקדקדים בצד אחד מייצגים את מספרי השורות, בצד השני את מספרי העמודות, וקשת בין שורה לעמודה מייצגת פיקסל בתמונה).
כל התוצאות המתוארות בפרק זה נכונות למודל *הצפוף*, שבו גרפים מיוצגים על ידי פונקציות $G: \binom{[n]}{2} \to \{0,1\}$.

במאמר המקורי של גולדרייך, גולדווסר ורון [83] הוכח כי את כל התכונות הניתנות לייצוג כ"תכונות חלוקה" בגרפים, כגון k-צביעות, או הכלה של תת גרף שלם גדול, ניתן לבדוק באמצעות מספר קבוע של דגימות (ראו גם [86]). אלון, פישר, קריבלביץ' וסגדי [7] הוכיחו כי התכונה של אי הכלת תת גרף מושרה F ממשפחה "אסורה" $\mathcal{F}$ (ללא חשיבות לסדר בין הקדקדים) ניתנת גם היא לבדיקה במספר קבוע של דגימות עבור כל משפחה סופית $\mathcal{F}$. תוצאתם נבעה ממשפט קומבינטורי, *למת ההסרה המושרית*, שהיא הכללה של למת ההסרה הידועה לגרפים של רוז'ה וסמרדי [5, 131]. למת ההסרה המושרית טוענת כי לכל משפחה סופית $\mathcal{F}$ כנ"ל ולכל $\varepsilon > 0$ קיים מספר

הקדמה

התפוצצות המידע בתחומי המדע וההנדסה בעידן הנוכחי העלתה את הצורך לפיתוח שיטות יעילות להבנה וניתוח מידע, ובפרט כאשר הגישה למידע מוגבלת. התחום של בדיקת תכונות מדגמית (Property Testing) חוקר נושא זה בדיוק – מה ניתן ומה לא ניתן להבין מדגימות קטנות מהמידע (שלעתים עשויות להיות אדפטיביות, כלומר כאלו שנבחרו בצורה המותאמת למידע הנצפה). מאז ייסודו של התחום לפני כעשרים וחמש שנים, על ידי רובינפלד וסודן [122] וכן גולדרייך, גולדווסר ורון [83], הוא נהנה מפריצות דרך משמעותיות, הן בשל התקדמות בהבנה המתמטית (קומבינטורית, אלגברית או טופולוגית) של התחום, והן בשל התקדמויות אלגוריתמיות מרשימות, שהסתמכו במקרים רבים על ההתקדמות המתמטית. למספר מקורות כלליים בנושא בדיקת תכונות, ראו [80, 81, 119, 120].

פורמלית, הבעיה הטיפוסית בבדיקת תכונות מדגמית היא מהצורה הבאה: בידינו היכולת לדגום מידע, המיוצג כפונקציה $f\colon X \to Y$ לא ידועה בעלת התחום X והטווח Y (שניהם ידועים), כאשר דגימה הינה שאילתא מהצורה $f(x)$ עבור x לבחירתנו. המטרה היא להחליט, בהסתברות טובה, האם f מקיימת תכונה מסוימת $\mathcal{P}$ ידועה מראש, או רחוקה מלקיים תכונה זו. עבור פרמטר מרחק $\varepsilon > 0$, אנו אומרים כי פונקציה היא ε-רחוקה מקיום התכונה אם יש לשנות את ערכי הפונקציה עבור לפחות $\varepsilon|X|$ קלטים על מנת שתקיים את התכונה. יעילות נמדדת בדרך כלל במספר הדגימות, ולפעמים גם בזמן הריצה הכולל של האלגוריתם. ככלל, אלגוריתמי בדיקת תכונות הינם תת-לינאריים, ואינם קוראים את הקלט כולו. לכן, האלגוריתם בהכרח הסתברותי, ותשובותיו נכונות בהסתברות טובה, אך לא באופן דטרמיניסטי.

באופן כללי, אלגוריתמי בדיקת תכונות נוטים להיות פשוטים יותר לניתוח כאשר המידע סימטרי יחסית. תופעה זו נצפתה במגוון תתי תחומים של בדיקת תכונות. לדוגמה, תכונות סימטריות של התפלגויות (כגון האנטרופיה או המרחק מאחידות) הן בדיוק אותן התכונות המוגדרות באופן מלא על ידי "טביעת האצבע" של ההתפלגות – זוהי ההיסטוגרמה של ההיסטוגרמה של ההתפלגות. אפיון זה הוביל להבנה מצוינת, יחסית, של יעילות אלגוריתמי בדיקת תכונות התפלגות סימטריות [135]. בגרפים (בפרט במודל ה"צפוף", בו גרפים מיוצגים על ידי מטריצת שכנויות), עד כה ההתקדמות בהבנת אלגוריתמי בדיקת תכונות התרחשה כמעט אך ורק עבור תכונות שבהן סדר הקדקדים אינו משנה. בבעיות בדיקת תכונות אלגבריות, סימטריות ושמורות (אינווריאנטים) נחקרו בהעמקה [129], ובעיות בעלות סימטריה רבה יותר נתפסות כ"פשוטות יותר" להבנה באופן כללי.

לעומת זאת, בבעיות בדיקת תכונות במידע לא סימטרי, כמו מידע סדרתי (למשל טקסט, סדרות עתיות, או מידע ביולוגי), תמונות, ופנקציות בוליאניות ממימד גבוה, ההתקדמות איטית בהרבה. תוצאות חזקות וכלליות למקרים בהם אין סימטריה נדירות יחסית, ורוב ההתקדמות המחקרית התמקדה בבעיות סדורות ספציפיות, כגון בדיקת מונוטוניות בפונקציות בוליאניות, או קמירות וקשירות בתמונות.

מבלי להסתכל על המידע) הינו $\Theta\left((\log n)^{\lfloor \log_2 k \rfloor}\right)$, כאשר n הינו אורך המידע כולו ו-k הוא אורך התבנית. בנוסף, אנו מוכיחים חסמים תחתונים עבור אלגוריתמים לא אדפטיביים לתבניות לא מונוטוניות, המראים כי השיפור שמשיגים אלגוריתמים כאלו בהשוואה לדגימה אקראית מתוך המידע הינו מזערי עבור רובן המוחלט של התבניות.

בחלקה השלישי של התזה, אנו עוסקים בבעיות מקומיות במידע מובנה. בהקשר זה, בעיה נקראת מקומית אם אפשר לתארה על ידי אוסף של תבניות רצופות קטנות "אסורות". הגדרה זו מתאימה לרבות מהבעיות שנחקרו בתחום בדיקת התכונות במבנים סדורים. התוצאה המרכזית מראה כי כל התכונות המקומיות במידע סדור ניתנות לבדיקה באמצעות מספר תת-לינארי של דגימות. אלגוריתם הבדיקה הכללי דוגם מבנים דמויי קליפה כדורית במידע. האלגוריתם הינו אופטימלי עבור מספר בעיות שנחקרו בעבר, במיוחד כאשר המידע חד-מימדי. במימד גבוה, זוהי התוצאה הראשונה המראה כי תכונות כגון קמירות או תת-מודולריות ניתנות לבדיקה באמצעות מספר תת-לינארי של דגימות.
בנוסף, אנו מוכיחים "למת שינוי" קומבינטורית המאפיינת את המבנה של תכונות סדורות מקומיות, ומשתמשים בה לבניית אלגוריתם בדיקה יעיל עבור בעיות חיפוש תבניות. בכך אנו עונים על שאלה שנשאלה על ידי פישר ונוימן ב-2001, באשר לקיומם של אלגוריתמים יעילים כאלו.

תקציר

תחום המחקר של בדיקת תכונות מדגמית (Property Testing) עוסק בשאלה הבאה: מה ניתן להסיק מתוך מידע נתון באמצעות מספר קטן של דגימות למידע זה. בעשרים וחמש השנים האחרונות, הושגו פריצות דרך רבות ומשמעותיות בתחום. עם זאת, רובן הגדול הושגו תחת ההנחה שלמידע יש מבנה פשוט יחסית (לדוגמה, כאשר המידע מוגרל מהתפלגות כלשהי), או שהבעיה הנדונה היא סימטרית בהוויתה (לדוגמה, המחקר בנושא בדיקת תכונות בגרפים עסק, רובו ככולו, בתכונות שמניחות שאין סדר מובנה בין הקדקדים השונים בגרף).

בתזה זו אנו עוסקים במגוון בעיות בבדיקת תכונות מדגמית שבהן המידע הינו בעל מבנה מורכב יחסית. בפרט, אנו מציגים מספר תופעות קומבינטוריות מפתיעות, שהבנתן מובילה לפיתוח שיטות עבודה חדשות במחקר בעיות מבניות. השימוש בשיטות אלו מוביל לפתרון או להתקדמות המחקר בכמה מהשאלות המרכזיות בתחום זה.

תזה זו מחולקת לשלושה חלקים. בחלקה הראשון, אנו עוסקים בבדיקת תכונות בגרפים סדורים (כלומר, גרפים להם סדר מובנה על הקדקדים). תוצאה ידועה של אלון ושפירא מראה כי כל תכונה תורשתית של גרפים לא סדורים, כלומר, גרפים בהם הסדר בין הקדקדים חסר משמעות, ניתנת לבדיקה באמצעות מספר קבוע של דגימות. עם זאת, הוכחת הטענה מסתמכת באופן משמעותי על הסימטריה הנובעת מההנחה שהקדקדים אינם סדורים, ואינה מתאימה עבור גרפים שקדקדיהם מסודרים במבנה מסוים (לדוגמא בתמונות, שניתן להציגן כגרף דו צדדי סדור, שמיקומי השורות והעמודות בו משמעותיים, כיוון שהם מייצגים מיקומי פיקסלים). אלון, פישר ונוימן שאלו ב-2007 האם ניתן להוכיח תוצאה ברוח דומה לזו הנזכרת לעיל עבור תמונות. אנו עונים על שאלה זו בחיוב, ובתוך כך מפתחים כלים מבוססי רגולריות-סמרדי, המתאימים למבנים סדורים.

בחלקה השני של התזה, אנו עוסקים במציאת תבניות גלובליות במידע סדרתי. בהנתן סדרה של מספרים ממשיים המכילה מופעים זרים רבים של תבנית סדר מסוימת, כיצד ניתן למצוא עותק יחיד של תבנית הסדר ביעילות? מסתבר כי לבעיה זו מבנה קומבינטורי מעניין ביותר, שהבנתו חיונית למציאת אלגוריתמים יעילים. באמצעות מחקר הבעיה הקומבינטורית, אנו פותרים שאלה פתוחה שהועלתה לאחרונה ע"י נוימן, רבינוביץ', רג'נדרפרסד וזולר. בנוסף, אנו פותרים במלואה את הבעיה הנ"ל במקרה בו התבנית הינה תת סדרה מונוטונית במקרים האדפטיבי והלא-אדפטיבי. באופן מפתיע, מספר הדגימות הנדרש עבור אלגוריתם לא אדפטיבי (שעליו "להכריז" על כלל הדגימות שתבוצענה מראש,

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

הפקולטה למדעים מדויקים
ע"ש ריימונד ובברלי סאקלר
ביה"ס למדעי המחשב ע"ש בלווטניק

אלגוריתמים מהירים לבעיות סדורות ומבניות

חיבור זה מוגש כחלק מהדרישות לקבלת תואר "דוקטור לפילוסופיה"

מאת

עומרי בן אליעזר

בהנחיית פרופ' נוגה אלון

יוני 2020