On Testing Graph Isomorphism and Group Properties

by
Laliv Tauber

Advisor: Prof. Oded Goldreich

Thesis for the degree
Master of Science

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

Submitted to the Scientific Council of the Weizmann Institute of Science
Rehovot, Israel
February 2024

Preface

We consider several testing problems. The first problem is testing graph isomorphism in the bounded-degree graph model. We focus on the case where one of the graphs is fixed. Our main result is that, for almost all d-regular n-vertex graphs H, testing isomorphism to H (the fixed graph) can be done using $\widetilde{O}(\sqrt{n})$ queries. This result is shown to be optimal (up to a polylog factor) by a matching lower bound, which also holds for almost all graphs H.

We also consider the graph isomorphism problem in the specific case where the degree bound is two. This leads to a general tester for every graph property, in this special case. The query complexity of the tester is $\text{poly}(1/\epsilon)$, which means it depends only on the proximity parameter ϵ.

In addition, we consider the topic of testing group properties. We focus on the problem of testing whether a binary operation over S is a group operation. Our main result is a tester of running-time $\widetilde{O}(|S|)$, which improves over the previously known tester that has running time $\widetilde{O}(|S|^{3/2})$. This tester leads easily to a tester for abelian groups with the same running time. We also show a lower bound of $\Omega(\log|S|)$ for testing each of the properties of being a group, an abelian group, or a cyclic group.

CONTENTS

CHAPTER 1

INTRODUCTION

This chapter is dedicated to a short presentation of the area of property testing, as well as a brief description of each of the chapters and its main results. The chapters are self-contained and can be read independently.

1.1 The field of Property Testing

Property testing is a field primarily concerned with the design of super-fast algorithms for approximate decision-making, where the decision refers to properties or parameters of huge objects. The term "approximate decision" refers to distinguishing between objects that have some property and objects that are far from having the property, where having a property means being in some set $\Pi = \cup_{n \in N} \Pi_n$ where Π_n is a set of n-long sequences over some predetermined alphabet denoted R_n, and being far from a property means being far from any object having the property.

We model testers as probabilistic oracle machines that access their main input (the object being tested) via queries to a function representing the object, and outputs 0 or 1 according to its decision. The object is usually represented by a function $f : [n] \rightarrow R_n$, and when the tester queries about $i \in [n]$ it is answered with the value $f(i)$. The tester is also given as inputs n and ϵ that represent the length of the object and the distance that the tester should work with respect to, respectively. We define the distance between two objects of length n to be the relative distance between the functions representing them (that is, the fraction of locations on which they disagree). We will say an object of length n is ϵ-far from a property if the distance between it and any object in Π_n is at least ϵ, and we will say it is ϵ-close otherwise.

Definition 1.1.1 (A tester for property Π): *A tester for property $\Pi = \cup_{n \in N} \Pi_n$ such that Π_n is a subset of the functions from $[n]$ to R_n is a probabilistic oracle machine that, on input parameters n and ϵ, and oracle access to a function $f : [n] \rightarrow R_n$ distinguishes between the following two cases.*

f represents an object in the property Π: *f is in Π_n.*

f is ϵ-far from representing an object in the property Π: *For every $g : [n] \rightarrow R_n$ such that $g \in \Pi_n$, it holds that f is ϵ-far from g.*

That is, the tester accepts any function in the property with probability at least $2/3$, and rejects any function that is ϵ-far from the property with probability at least $2/3$. The tester is said to have one-sided error *if it always accepts any function in the property.*

Usually, the main focus will be on the query complexity of the tester as a function of n and ϵ, but the time complexity is also important, especially when the query complexity is not independent of n.

1.2 Testing graph isomorphism in the bounded-degree model

In Chapter 2 we consider the problem of testing graph isomorphism in the bounded-degree graph model. We focus on the fixed graph version where the task is testing whether an (n-vertex) graph G is isomorphic to a fixed (n-vertex) graph H (which "massively parametrized" the property). In contrast, in the two input-graphs version, the input is a pair of (n-vertex) graphs, and the task is testing whether they are isomorphic.

The most relevant previous result is in [13], which showed a lower bound of $\widetilde{\Omega}(n^{1/2})$ queries for the fixed graph version, using graphs that consist of small connected components. We show that this lower bound holds for almost all fixed regular graphs and not only for graphs made out of small components. In addition, we show a tester that uses $\widetilde{O}(\sqrt{n})$ queries and works for almost all regular n-vertex graphs. The combination of these two results implies that our tester is optimal (up to a polylog factor) for almost all d-regular n-vertex graphs H. These two results are stated in the following theorems.

Theorem 1.2.1 (a tester of isomorphism to a fixed random regular graph): *In the bounded-degree graph model with degree bound $d \geq 3$, for almost all d-regular n-vertex graphs H, and for every $\epsilon > 0$, there exists an ϵ-tester of isomorphism to H that makes $\widetilde{O}(n^{1/2}/\epsilon)$ queries. Furthermore, the tester has one-sided error.*

Theorem 1.2.2 (lower bound for testing isomorphism to a fixed random regular graph): *In the bounded-degree graph model with degree bound $d \geq 3$, for almost all d-regular n-vertex graphs H, testing isomorphism to H requires $\Omega(n^{1/2})$ queries.*

This lower bound holds also for two-sided errors testers.

1.3 Testing graph properties when the degree bound is 2

Although testing graph isomorphism in the case where $d \geq 3$ seems to be not such an easy task, when $d = 2$ things are different. In that case, any graph is just a collection of paths and cycles, and testing isomorphism becomes quite simple.

Considering the above, we associate any graph with a profile that essentially represents the number of paths and cycles of each length (up to $O(1/\epsilon)$) in the graph. It is easy to see that the distance between two graphs is straightly related to the difference between their two profiles. These observations, along with some improvements, leads to the main result of Chapter 3.

Theorem 1.3.1 (testing any graph property): *Let Π be an arbitrary graph property. Then, in the bounded-degree graph model with degree bound 2, the property Π can be tested using $\widetilde{O}(1/\epsilon^2)$ queries.*

1.4 Testing group properties

In Chapter 4 we consider the problem of testing whether a given function $f : [n] \times [n] \to [n]$ represents a group operation. As we show in Chapter 4, testing the above with query complexity of $\widetilde{O}(n/\epsilon)$ is quite simple and derived from a generic tester to any set of functions. The problem is that the time complexity of this tester is exponential in the query complexity. In an attempt to overcome this problem, we present two more testers, and our main result is a tester with both query complexity and time complexity of $\widetilde{O}(n/\epsilon)$. This tester can be easily adapted to testing also whether f represent an abelian group. Note that we present the complexities as a function of n, although the size of the input is n^2.

Theorem 1.4.1 (efficiently testing the property of being a group): *There exists a $\widetilde{O}(n/\epsilon)$-time tester for the property of being a group. Furthermore, the tester has one-sided error.*

Theorem 1.4.2 (efficiently testing the property of being an Abelian group): *There exists a $\widetilde{O}(n/\epsilon)$-time tester for the property of being an Abelian group. Furthermore, the tester has one-sided error.*

Our tester uses the fact that f represents a group if and only if it is both cancellative and associative. In order to keep the complexities almost linear, we need to test these properties in a subtle way that allows us to define a self-corrector for f in a way that serves us.

In addition, we show a lower bound on the query complexity for testing any group property that contains all cyclic groups of prime order.

Theorem 1.4.3 (a lower bound on testing properties of groups): *Let $\mathcal{G}$ be a set of finite groups that is closed under isomorphism and contains all cyclic groups of prime order. Then, testing $\mathcal{G}$ requires $\Omega(\log n)$ queries.*

This implies that testing each of the properties of being a group, an abelian group or a cyclic group requires $\Omega(\log n)$ queries.

CHAPTER 2

TESTING GRAPH ISOMORPHISM IN THE BOUNDED-DEGREE MODEL

2.1 Introduction

In this chapter we consider the problem of testing graph isomorphism In the bounded-degree graph model.

Recall that the graphs $G_1 = ([n], E_1)$ and $G_2 = ([n], E_2)$ are isomorphic if there exists a bijection $\pi : [n] \to [n]$, called an isomorphism, such that $\{\pi(u), \pi(v)\} \in E_2$ if and only if $\{u, v\} \in E_1$; in this case, we may write $G_2 = \pi(G_1)$. Needless to say, graph isomorphism is one of the most basic notions regarding graphs. In fact, it is the basis for the notion of a graph property (i.e., a set of graphs that is closed under isomorphism), which reflects the interest in actual structural features of the graph while ignoring the labeling of its vertices (and edges).

2.1.1 Testing in the bounded-degree graph model

In the bounded-degree graph model, graphs are represented by their incidence functions and distances between graphs are measured accordingly. Specifically, for a fixed degree bound d, a graph $G = ([n], E)$ of maximum degree at most d is represented by a function $g : [n] \times [d] \to [[n]]$, where $[[n]] = \{0, 1, ..., n\} = [n] \cup \{0\}$, such that $g(v, i) = u \in [n]$ if u is the i^{th} neighbor of v (in G), and $g(v, i) = 0$ if v has less than i neighbors. (Note that this representation is not unique, since the order of the edges incident at each vertex is unspecified by the graph itself.)

The graph $G = ([n], E)$ is said to be ϵ-far from the graph $G' = ([n], E')$ if the symmetric difference between E and E' is larger than $\epsilon dn/2$ (equiv., if any representations $g : [n] \times [d] \to [[n]]$ and $g' : [n] \times [d] \to [[n]]$ of G and G' differ on more than ϵdn entries (i.e., $|\{(v, i) : g(v, i) \neq g'(v, i)\}| > \epsilon dn$)). Otherwise, the graphs are ϵ-close. The graph $G = ([n], E)$ is said to be ϵ-far from a graph property Π if G is ϵ-far from any graph in Π.

Fixing d, n and ϵ, we say that an oracle machine is an ϵ-tester of Π if, when given oracle access to an incidence function of an n-vertex graph (of maximum degree d), it distinguishes between the case that the graph is in Π and the case that the graph is ϵ-far from Π; that is, the tester accepts with probability at least $2/3$ in the first case and rejects with probability at least $2/3$ in the second case. If the tester always accepts any graph in Π, we say that it has one-sided error; otherwise, we say that it has two-sided error.

Indeed, testing isomorphism to a fixed graph H is the task of testing the property that consists of the set of graphs that are isomorphic to H (i.e., H is a massive parameter that specifies the property).[1] In this case, the tester is given oracle access to the (incidence function) of a graph and is required to determine

[1]See [11, Sec. 12.7.2] for a brief discussion of "massively parametrized" properties.

whether this graph is isomorphic to H. (Testing isomorphism between a pair of input graphs is formulated by an extension of the model that refers to machines that are given access to two oracles rather than to one oracle, where each oracle represents the incidence function of the corresponding graph.)

The complexity of testing a graph property Π (in the bounded degree graph model) is measured in terms of the degree bound d, the number of vertices n, and the proximity parameter ϵ. Typically, the degree bound is a constant, and the dependency on it is ignored. Furthermore, when discussing lower bounds, we ignore the dependency on the proximity parameter, which is assumed to be a (sufficiently small positive) constant. That is, saying "testing Π requires Q queries" means that *for some $\epsilon > 0$, any ϵ-tester of Π requires Q queries.*

We stress that throughout the text, the number of vertices, denoted n, is viewed as a varying parameter, and complexities are always stated as a function of n. In particular, the aforementioned "fixed (n-vertex) graph" H should be viewed as a parameter, which is explicitly given to the potential testers (just as n).

2.1.2 The most relevant previous results

The current work is most related to [13], which presented non-trivial lower bound on the complexity of testing isomorphism in the bounded-degree model. The focus of [13] was on the special case in which the graphs consist of small connected components; that is, n-vertex graphs with connected components of size $\text{poly}(\log n)$. Ignoring the dependence on the proximity parameter, the main results presented in [13] are:

1. The query complexity of testing isomorphism to a fixed n-vertex graph (with connected components of size $\text{poly}(\log n)$) is $\widetilde{\Theta}(n^{1/2})$.

2. The query complexity of testing isomorphism between two n-vertex graphs (with connected components of size $\text{poly}(\log n)$) is $\widetilde{\Theta}(n^{2/3})$.

These results were proved by relating these testing problems to analogous problems about equality between multi-sets (containing $o(n)$ elements of $[n]$), whereas the latter problems were related to analogous problems about distribution testing.

Focusing on the fixed graph version, we ask *how does the query complexity of testing isomorphism depend on the structure of the fixed graph.* We interpret the foregoing result of [13] as providing an answer to the special case of the class of fixed graphs that consist of *small connected components.* Needless to say, this is not a very natural class of graphs. Furthermore, as admitted in [13], the analysis of this class of fixed graphs reduces to the analysis of multi-sets that merely reflect the statistics of the different types of connected components.

2.1.3 New results

Our results addresses the foregoing question. Specifically, we show that the lower bounds proved in [13] holds for almost all fixed regular graphs rather than only for fixed graphs with small connected components. Most importantly, we present a tester that uses $\widetilde{O}(\sqrt{n})$ queries and works for almost all regular n-vertex graphs. More precisely –

Theorem 2.1.1 (a tester of isomorphism to a fixed random regular graph): *In the bounded-degree graph model with degree bound $d \geq 3$, for almost all d-regular n-vertex graphs H, and for every $\epsilon > 0$, there exists an ϵ-tester of isomorphism to H that makes $\widetilde{O}(n^{1/2}/\epsilon)$ queries. Furthermore, the tester has one-sided error.*

Interestingly, the foregoing result is tight in the following sense.

Theorem 2.1.2 (lower bound for testing isomorphism to a fixed random regular graph): *In the bounded-degree graph model with degree bound $d \geq 3$, for almost all d-regular n-vertex graphs H, testing isomorphism to H requires $\Omega(n^{1/2})$ queries.*

We stress that the aforementioned lower bound refers also to two-sided error testers. We mention that it is easy to prove the *existence* of d-regular n-vertex graphs H such that testing isomorphism to H *with one-sided error* requires $\Omega(n)$ queries [13, Thm. 2.6]. On the other hand, the requirement $d \geq 3$ is essential, since when $d \leq 2$ every graph property can be ϵ-tested using $\widetilde{O}(1/\epsilon^2)$ queries [17].

Needless to say, there may be fixed d-regular n-vertex graphs such that testing isomorphism to them requires $\Omega(n^c)$ queries, for some constant $c > 1/2$ (and maybe even for all constants $c < 1$); but Theorem 2.1.1 asserts that such graphs, if they exist, are quite rare. Likewise, Theorem 2.1.2 asserts that fixed d-regular graphs that allow for more efficient testing of isomorphism to them are also quite rare.

Actually, both theorems are special cases of more general statements, which identify structural properties of the fixed graph H such that (1) the claimed complexity bounds hold for any fixed graph that satisfies these properties, and (2) these properties hold for random regular graphs. For example, the tester that underlies the proof of Theorem 2.1.1 works well for every fixed n-vertex graph H that has logarithmic diameter and "different $\widetilde{O}(n^{1/2})$-sized neighborhoods" (i.e., a BFS that stops after encountering $\widetilde{O}(n^{1/2})$ vertices sees non-isomorphic subgraphs when started at different vertices of H). The "unique neighborhood" condition is stated at the beginning of Section 2.1.4.1, and a generalization of it is captured by Definition 2.2.5. A condition that suffices for the lower-bound of Theorem 2.1.2 is stated in Definition 2.3.1.

2.1.4 Proof sketches

As stated above, we identify sufficient conditions on the fixed n-vertex graph H such that testing isomorphism to H has query complexity $\widetilde{\Theta}(n^{1/2})$. The sufficient conditions used for the upper and lower bounds are not identical, but both conditions hold for almost all regular graphs. Furthermore, weaker quantitative versions of these conditions do yield meaningful (alas weaker) corresponding bounds (see Theorem 2.2.6 and Corollary 2.3.4, resp.).

2.1.4.1 The tester: Proof sketch for Theorem 2.1.1

The key observation is that, for $\ell^* = \log_{d-1} \widetilde{O}(n^{1/2})$ (equiv., $d \cdot (d-1)^{\ell^*-1} = \widetilde{O}(n^{1/2})$), the ℓ^*-neighborhoods of vertices in a random d-regular n-vertex graph H are unique, where the ℓ-neighborhood of a vertex v in a graph is the subgraph induced by the set of vertices that are at distance at most ℓ from v in the graph. In other words, as proved by Mossel and Sun [23], *in a random d-regular n-vertex graph, the ℓ^*-neighborhoods of vertices are pairwise non-isomorphic.* Our tester works for any fixed d-regular n-vertex graph H of logarithmic diameter in which the ℓ^*-neighborhoods of vertices are pairwise non-isomorphic. Indeed, the additional requirement of having logarithmic diameter is also satisfied by a random regular graph.

For any fixed regular graph H that satisfies the foregoing conditions, testing whether an input graph G is isomorphic to H reduces to selecting a random edge in H and checking that the two vertices of G that correspond to its endpoints are adjacent in G, where the correspondence means *having isomorphic ℓ^*-neighborhoods.* Note that, since the ℓ^*-neighborhoods in H are distinct, the foregoing correspondence constitutes a one-to-one mapping of vertices of H to vertices of G. Hence, G is isomorphic to H if and only if each edge of H is mapped (by this correspondence) to an edge of G.

A key issue, which was ignored so far, is *finding in G a vertex that corresponds to a given vertex u in H.* (The following description refers to the case that G is isomorphic to H; it may fail otherwise, and such a failure indicates that G is not isomorphic to H.) Here we rely on the fact that H has logarithmic diameter. We start by selecting an arbitrary (start) vertex v_0 in G, and then "locate" the "copy" of v_0 in H by exploring v_0's ℓ^*-neighborhood (in G); that is, *the ℓ^*-neighborhood of v_0 in G determines the unique vertex u_0 in H that has an isomorphic ℓ^*-neighborhood.* Next, we determine a short path in H leading from u_0 to u. Denoting this path by $(u_0, u_1, ..., u_\ell = u)$, we find the corresponding vertices in G in ℓ iterations. In the i^{th} iteration, having found (in G) the vertex v_{i-1} that corresponds to u_{i-1}, we explore the ℓ^*-neighborhoods of each neighbor of v_{i-1} in G and determine which of these neighbors corresponds to u_i. That is, we start by locating in H an arbitrary vertex of G, denoting this location (in H) by u_0, and then iteratively locate in G each vertex of H that is on the short path (in H) from u_0 to the desired vertex u_ℓ.

The actual tester. For any fixed graph H that satisfies the foregoing conditions, our tester proceeds as follows. It selects uniformly at random $t = O(1/\epsilon)$ edges in the fixed graph H, and tries to locate the endpoints of these edges (of H) in the input graph G. Next, the tester checks that each vertex-pair in G that corresponds to a chosen edge in H is indeed adjecent in G, where the correspondence is defined by the foregoing locating process. Needless to say, if the tester failed to find (or rather uniquely determine) in G a vertex that corresponds to any of the $2t$ selected vertices of H, then it rejects. Ditto if any of the pairs corresponding to the endpoints of these edges (of H) are not adjecent in G.

Clearly, this algorithm always accepts a graph G that is isomorphic to H. On the other hand, let U denote the set of vertices in H that are properly located in G (by the foregoing "locating" procedure), and let $\mu : U \to [n]$ denote the locating mapping (from H to G). Letting E' denote the set of edges of H such that their endpoints are both in U and their μ-images are adjacent in G, observe that if G is accepted with probability at least $1/3$, then it must be that $|E'| \geq (1 - 0.5 \cdot \epsilon) \cdot dn/2$. In this case, arbitrarily extending μ to a bijection from $[n]$ to $[n]$, it follows $\mu(H)$ is ϵ-close to G.

Digest. The tester relies on the fact that, for a random n-vertex regular graph H, if the input graph G is isomorphic to the fixed graph H, then it is possible to locate vertices of G in H by making $(d-1)^{\ell^*} = \widetilde{O}(\sqrt{n})$ queries. Using this fact, the tester locates vertices of H in G, by finding a short path to the desired location in G using a corresponding short path in H. This is akin the proof of [18, Thm. 4.9]; see further discussion in Section 2.2.

The fact that the tester relies on locating random (pairs of adjacent) vertices of H in the input graph G rather than on locating random vertices of G in H is crucial. A mapping of vertices of H to vertices of G that "preserves ℓ^*-neighborhoods" must be one-to-one, since the vertices of H have different ℓ^*-neighborhoods, whereas an analogous mapping of vertices of G to vertices of H may be many-to-one. Consider, for example, the case that G consists of two copies of the same $n/2$-vertex graph, denoted G', whereas H consists of a copy of G' and some other $n/2$-vertex graph (such that all ℓ^*-neighborhoods are distinct).[2]

2.1.4.2 The lower bound: Proof sketch for Theorem 2.1.2

We shall prove that, for almost all pairs of d-regular n-vertex graphs, H and G, when explicitly given both H and G, distinguishing between a random isomorphic copy of H and a random isomorphic copy of G requires $\Omega(n^{1/2})$ queries. Intuitively, this is the case because a $o(n^{1/2})$-step exploration of either graphs is unlikely to encounter a cycle, and so each exploration will just see the forest that is spanned by its queries. Observing that, for any fixed n-vertex graph H, a random d-regular n-vertex graph G is $\Omega(1)$-far from being isomorphic to H, it follows that, for almost all d-regular graphs H, testing isomorphism to H requires $\Omega(n^{1/2})$ queries.

Actually, the indistinguishablity claim holds for random isomorphic copies of any two d-regular n-vertex graphs H and G such that both $\mathtt{sc}(H)$ and $\mathtt{sc}(G)$ are $O(1/n)$, where $\mathtt{sc}$ is as defined in [14, Def. 3.2.1] (reproduced as Definition 2.3.1). The fact that a $o(n^{1/2})$-step exploration of such a graph is unlikely to find a cycle is established implicitly in the proof of [14, Lem. 3.2.2], whereas the fact that random d-regular n-vertex graphs have $\mathtt{sc}$-value $O(1/n)$ is proved in [14, Lem. 3.2.3].

2.1.5 Other related work

The two versions of the graph isomorphism testing problem were considered before [9, 20, 25], in various models. Among these studies, the work of Newman and Sohler [25] is most relevant to us, since it is in the bounded-degree graph model; but, like [13] (which was reviewed in Section 2.1.2), their work refers to a restricted class of graphs. Specifically, Newman and Sohler focus on testing arbitray properties of *hyperfinite graphs* (in the bounded-degree graph model). Towards that end, they proved that testing isomorphism between two hyperfinite graphs has complexity that only depends on the proximity parameter (i.e., ϵ); see [25, Thm. 3.2]. Loosely speaking, hyperfinite graphs are close to graphs that consists of connected components of constant size, where the level of proximity is the function of the latter constant.

[2]One can modify this example to obtain connected graphs by adding a single edge between the two $n/2$-vertex subgraphs.

A few years earlier, both testing problems were studied by Fischer and Matsliah [9] *in the dense graph model* (reviewed in [11, Chap. 8]). Interestingly, in all cases they considered, the complexity is sublinear (in the number of vertex-pairs), but is a constant power of that number. In particular, isomorphism between two n-vertex input graphs can be tested with one-sided error using $\widetilde{O}(n^{3/2})$ queries, and (two-sided error) testing of isomorphism to some fixed n-vertex graphs requires $\widetilde{\Omega}(n^{1/2})$ queries.

More recently, these testing problems were studied by Kusumoto and Yoshida [20] *in the general graph model* (reviewed in [11, Chap. 10])). They considered the case that the graphs are promised to be forests (or, alternatively, the property requires the graphs to be forests), and showed that in this case the query complexity is polylogarithmic in the size of the graph.

2.1.6 Organization

Section 2.2 contains the proof of Theorem 2.1.1. Using the fact, proved by Mossel and Sun [23], that the unique neighborhoods condition holds in random regular graphs, we provide a more detailed description of the tester and its analysis.

Section 2.3 provides a proof of Theorem 2.1.2, which establishes a query complexity lower bound that matches the upper bound provided by the tester (upto a polylog factor). This proof combines a standard (lower bound) technique with ideas and results from [14, Sec. 3.2].

2.2 The tester: Proof of Theorem 2.1.1

As stated in Section 2.1.4.1, our tester for isomorphism to a fixed n-vertex graph H relies on the hypothesis that the $(\log_{d-1}\widetilde{O}(n^{1/2}))$-neighborhoods of vertices in H are unique (i.e, these neighborhoods are non-isomorphic). Recall that the ℓ-neighborhood of a vertex v in a graph is the subgraph induced by the set of vertices that are at distance at most ℓ from v in the graph. For a (bounded-degree) graph and $\ell \in \mathbb{N}$, the unique ℓ-neighborhoods condition asserts that the ℓ-neighborhoods of the vertices of the graph are non-isomorphic. We shall use the following result of Mossel and Sun [23].

Lemma 2.2.1 (unique neighborhoods in random graphs [23, top p. 2]): *For a constant $d \geq 3$ and varying $n \in \mathbb{N}$, let $\ell^* = \log_{d-1}\widetilde{O}(n^{1/2})$. Then, in a random d-regular n-vertex graph, with probability $1-o(1)$, the ℓ^*-neighborhoods of the n vertices in the graph are pairwise non-isomorphic.*

Actually, the result proved in [23] holds for $\ell^* = \log_{d-1} O(n \log n)^{1/2}$. In contrast, for $\ell' = \log_{d-1} o(n^{1/2})$, the ℓ'-neighborhoods of most pairs of vertices in a random d-regular n-vertex graph are isomorphic; actually, they are both trees (of depth ℓ'). This follows as a special case of Lemma 2.3.2 (combined with [14, Lem. 3.2.3]).

The tester. For any fixed graph H of logarithmic diameter that satisfies the unique neighborhoods condition, our tester proceeds as outlined in Section 2.1.4.1. For sake of good order, we shall detail this tester below. We stress that the tester has free access to the fixed graph H and is given query access to the incidence function of the input graph G. (Hence, exploring the ℓ-neighborhood of a vertex in G requires $d \cdot (d-1)^{\ell-1}$ queries, whereas exploring the ℓ-neighborhood of a vertex in H requires no queries.) Recalling that our tester relies on a procedure for locating vertices of H in G, we detail this procedure first.

Algorithm 2.2.2 (locating vertices of a fixed graph in an input graph): *Suppose that H is a fixed d-regular n-vertex graph that has diameter $D = O(\log n)$ and satisfies the unique ℓ^*-neighborhoods condition, where $\ell^* = \log_{d-1}\widetilde{O}(n^{1/2})$. Then, given a vertex u in H and oracle access to an input graph G, which is allegedly isomorphic to H, we* locate u in G *as follows.*

1. *We select arbitrarily* (but deterministically) *a vertex v_0 in G, and locate it in H. This is done by exploring the ℓ^*-neighborhood of v_0 in G and finding a vertex u_0 in H that has an isomorphic ℓ^*-neighborhood.*

2. *We* (deterministically) *find a short path* (i.e., a path of length at most D) *in H leading from u_0 to u. Let us denote this path by $(u_0, u_1, ..., u_\ell)$, where $\ell \leq D$ and $u_\ell = u$.*

3. *For $i = 1, ..., \ell$, we locate u_i in G by exploring the ℓ^*-neighborhood* (in G) *of each neighbor of v_{i-1} in G and comparing it to the ℓ^*-neighborhood of u_i in H; that is, v_i is determined as the* unique *neighbor of v_{i-1} in G that has an ℓ^*-neighborhood* (in G) *that is isomorphic to the ℓ^*-neighborhood of u_i* (in H).

If any of the foregoing steps failed (i.e., either v_0 was not located in H or v_{i-1} has no neighbor or more than one neighbor that fits u_i)*, then we announce failure. Otherwise, we rule that $v = v_\ell$ is the vertex of G that corresponds to u.*

We stress that the foregoing algorithm is deterministic, and that it always locate any vertex u (of H) in any graph G that is isomorphic to H. The latter assertion is based on the hypothesis that H has diameter D and satisfies the unique ℓ^*-neighborhoods condition. The query complexity of Algorithm 2.2.2 is $D \cdot d \cdot (d \cdot (d-1)^{\ell^*-1}) = O(D \cdot (d-1)^{\ell^*})$, which is $\text{poly}(\log n) \cdot n^{1/2}$ when $\ell^* = \log_{d-1} \widetilde{O}(n^{1/2})$ and $D = \text{poly}(\log n)$. Using Algorithm 2.2.2, we finally detail our tester.

Algorithm 2.2.3 (tester of isomorphism for a fixed regular graph): *Suppose that H is a fixed d-regular n-vertex graph that has diameter $D = O(\log n)$ and satisfies the unique ℓ^*-neighborhoods condition, where $\ell^* = \log_{d-1} \widetilde{O}(n^{1/2})$. Then, given oracle access to an input graph G, we test whether G is isomorphic to H as follows.*

1. *We select uniformly at random $m \stackrel{\text{def}}{=} O(1/\epsilon)$ edges, denoted $\{r_1, s_1\}, ..., \{r_m, s_m\}$, in the fixed graph H.*

2. *For every $i = 1, .., m$, we locate r_i and s_i in the input graph G, by invoking Algorithm 2.2.2. If any of these invocations failed, we reject. Otherwise, we denote the corresponding locations in G by $\mu(r_i)$ and $\mu(s_i)$.*

3. *For every $i = 1, .., m$, we check whether $\mu(r_i)$ neighbors $\mu(s_i)$ in G, by querying the d incidences of $\mu(r_i)$. We accept if all these m checks were successful* (i.e., $\{\mu(r_i), \mu(s_i)\}$ is an edge in G for every $i \in [m]$)*, and otherwise we reject.*

Observe that Algorithm 2.2.3 has query complexity $O(\epsilon^{-1} \cdot (d-1)^{\ell^*} \cdot D)$, which is $\text{poly}(\log n) \cdot n^{1/2}/\epsilon$. Clearly, this algorithm always accepts a graph G that is isomorphic to H. We now show that if G is ϵ-far from being isomorphic to H, then the tester rejects with probability at least 2/3. We shall actually prove the contrapositive.

Claim 2.2.4 (on the graphs accepted by Algorithm 2.2.3): *If Algorithm 2.2.3 accepts G with probability at least* 1/3*, then G is ϵ-close to being isomorphic to H.*

Proof: Let U denote the set of vertices u in H on which Algorithm 2.2.2 does not fail. For each $u \in U$, let $\mu(u)$ denote the location of u in G as determined by Algorithm 2.2.2, and note that μ is injective since the ℓ^*-neighborhoods of vertices in H are pairwise non-isomorphic.[3]

Let $E' \subset \binom{U}{2}$ denote the set of edges of H such that their mapping under μ is an edge in G; that is, $\{r, s\} \in E'$ if $\{\mu(r), \mu(s)\}$ is defined and is an edge in G. Using the hypothesis that G is accepted with probability at least 1/3, it follows that $|E'| \geq (1 - 0.5 \cdot \epsilon) \cdot dn/2$. Extending μ arbitrarily to a bijection from the vertex set of H to the vertex set of G, and using the fact that at least $(1 - 0.5 \cdot \epsilon) \cdot dn$ of the incidences of G agree with those in $\mu(H)$, it follows that G is ϵ-close to $\mu(H)$. ■

Proof of Theorem 2.1.1. Recall that Lemma 2.2.1 asserts that almost all d-regular n-vertex graphs satisfy the unique ℓ^*-neighborhoods condition, and that the same holds regarding having logarithmic diameter. Using Algorithm 2.2.3 (as analyzed in Claim 2.2.4), we establish Theorem 2.1.1.

[3]In contrast, an analogue mapping from G to H is not necessarily injective, becuase the ℓ^*-neighborhoods of vertices in G are not necessarily pairwise non-isomorphic.

Abstraction and generalization. As stated in Section 2.1.4.1, our tester relies on the fact that if the input graph G is isomorphic to the fixed graph H, then it is possible to locate vertices of G in H by making $\widetilde{O}(\sqrt{n})$ queries to G. Using this fact, in this case, the tester (or rather Algorithm 2.2.2) locates vertices of H in G, by finding a short path to the desired location in G using a corresponding short path in H. Hence, the pivotal notion is of *locating vertices of an input graph G in a fixed graph H*, when G is isomorphic to H, by making relatively few queries to G. This notion, which (at least in its current form) is only relevant to asymmetric graphs, was introduced in [18, Sec. 4.4]. We review it next (following the more general formalism of [14, Def. 1.2]).

Definition 2.2.5 (local self-ordering procedures):[4] *For a function $q : \mathbb{N} \to \mathbb{N}$, a q-query* local self-ordering procedure *for an asymmetric graph $H = ([n], E)$ is a randomized oracle machine that, given a vertex v in any graph $G = ([n], F)$ that is isomorphic to H and oracle access to G, makes at most $q(n)$ queries, and outputs, with probability at least $2/3$, the vertex that corresponds to v in H; that is, it outputs $\phi(v) \in [n]$ for the unique bijection $\phi : [n] \to [n]$ such that $\phi(G) = H$* (i.e., the unique isomorphism of G to H).

Note that exploring the ℓ-neighborhood of v in G, in case H has unique ℓ-neighborhood, yields a deterministic $d\cdot(d-1)^{\ell-1}$-query local self-ordering procedure for H. Hence, Algorithm 2.2.2 can be generalized by using an arbitrary local self-ordering procedure for H (instead of the exploration of ℓ^*-neighborhoods). We mention that such an algorithm underlies the proof of [18, Thm. 4.9]. Plugging this algorithm in Algorithm 2.2.3, we get the following result.

Theorem 2.2.6 (generalization of Theorem 2.1.1): *Suppose that H is an asymmetric n-vertex graph of maximal degree d and diameter $D(n)$ that has a $q(n)$-query local self-ordering procedure. Then, in the bounded-degree graph model with degree bound d, for every $\epsilon > 0$, there exists an ϵ-tester of isomorphism to H that makes $\widetilde{O}(D(n)/\epsilon) \cdot q(n)$ queries. Furthermore, if the local self-ordering procedure is deterministic, then the tester has one-sided error and query complexity $O(D(n) \cdot q(n)/\epsilon)$.*

Proof: The furthermore claim (which refers to deterministic procedures) is proved by a straightforward implementation of the foregoing discussion. Specifically, starting with Algorithm 2.2.3, we replace the exploration of ℓ^*-neighborhoods used inside Algorithm 2.2.2 by the (guaranteed) deterministic local self-ordering procedure. Hence, when given oracle access to a graph that is isomorphic to H, this procedure always answers correctly, and we always accept. The analysis of the case that the input graph is not isomorphic to H relies on the fact that the (deterministic) local self-ordering procedure always yields the same answer (to the same input), and this feature is inherited by the revised Algorithm 2.2.2. At this point, the analysis proceeds as in the proof of Claim 2.2.4.

The foregoing fact no longer holds in the case that the local self-ordering procedure is randomized. In this case, the answer of the local self-ordering procedure may be wrong with probability at most 1/3 when the tested graph is isomorphic to H and may be arbitrarily distributed otherwise. Hence, we first reduce the error probability of the self-ordering procedure to $\epsilon' \stackrel{\text{def}}{=} o(\epsilon/D(n))$, which is obtained by using $O(\log(D(n)/\epsilon))$ invocations (and ruling by majority). The resulting procedure will be used inside Algorithm 2.2.2 (instead of the exploration of ℓ^*-neighborhoods). Hence, when Algorithm 2.2.2 is invoked on input v and oracle access to a graph that is isomorphic to H, it outputs the location of v in H with probability at least $1 - O(D(n)) \cdot \epsilon' = 1 - o(\epsilon)$. It follows that the revised Algorithm 2.2.3 accepts each graph that is isomorphic to H with probability at least $1 - m \cdot o(\epsilon) > 2/3$.

Lastly, we prove that if G is ϵ-far from being isomorphic to H, then it is rejected with probability at least 2/3. We again prove the contrapositive: Assuming that G is accepted with probability at least 1/3, we shall show that G is ϵ-close to being isomorphic to H. To simplify the analysis, we assume that all $O(D(n)/\epsilon)$ invocation of the local self-ordering procedure use the same random choices, while noting that this does not affect the analysis of the former case (i.e., of G being isomorphic to H), where we used a union bound

[4]The term self-ordering refers to the fact that, for a fixed (labeled) graph H, when given an unlabeled version of H, one can find the actual labels by looking at the unlabeled graph; in other words, given oracle access to $\pi(H)$, one can uniquely determine π (equiv., π^{-1}). In *local* self-ordering, when given v one is only required to find $\pi^{-1}(v)$ (equiv., given $\pi(u)$, one is required to find u).

on all $O((D(n)/\epsilon)$ invocations. Using an averaging argument, we fix a sequence of random choices for the local self-ordering procedure such that, when using the residual deterministic locating procedure, the revised Algorithm 2.2.3 accepts G with probability at least $1/3$. At this point the analysis proceeds as in the case that the local self-ordering procedure is deterministic. $\blacksquare$

2.3 Proof of the lower bound (Theorem 2.1.2)

Theorem 2.1.2 is proved by presenting, for each d-regular n-vertex graph H, two distributions on d-regular n-vertex graphs, denoted $\mathcal{D}_1$ and $\mathcal{D}_2$, such that *for almost all possible H's* the following holds:

1. With probability 1, a graph drawn from $\mathcal{D}_1$ is isomorphic to H.

2. With probability $1 - o(1)$, a graph drawn from $\mathcal{D}_2$ is $\Omega(1)$-far from being isomorphic to H.

3. No algorithm A_H that makes $o(n^{1/2})$ queries to its oracle, can distinguish between the case that the oracle is selected from $\mathcal{D}_1$ and the case that the oracle is selected from $\mathcal{D}_2$; that is, letting A_H^G denote the verdict of A_H when given oracle access to a graph G, it holds that

$$\left|\Pr_{G\sim\mathcal{D}_1}[A_H^G\!=\!1] - \Pr_{G\sim\mathcal{D}_2}[A_H^G\!=\!1]\right| = o(1). \tag{2.1}$$

   We stress that the algorithm A_H may depend arbitrarily on H.

Since a test should distinguish the two distributions (i.e., output 1 with probability at least $\frac{2}{3}$ on graphs drawn from $\mathcal{D}_1$ while outputting 1 with probability at most $\frac{1}{3} + o(1)$ on graphs drawn from $\mathcal{D}_2$), it follows that a tester must make $\Omega(n^{1/2})$ queries. We stress that the foregoing holds for almost all setting of the fixed d-regular n-vertex graph H.

In particular, we shall use the following two distributions: The distribution $\mathcal{D}_1$ is obtained by selecting a random isomorphic copy of H, and $\mathcal{D}_2$ is obtained by selecting uniformly at random a d-regular n-vertex graph. Actually, as stated in Section 2.1.4.2, we shall prove that for almost all pairs of d-regular n-vertex graphs, H and G, given H and G, it is hard to distinguish a random isomorphic copy of H from a random isomorphic copy of G, whereas G is $\Omega(1)$-far from being isomorphic to H.[5]

Recall that hardness to distinguish the foregoing two distributions refers to algorithms that make $o(n^{1/2})$ queries to the input graph (which is either a random isomorphic copy of H or a random isomorphic copy of G). Intuitively, the indistinguishablity claim is due to the fact that $o(n^{1/2})$-step exploration of either graphs is unlikely to encounter a cycle, and so such an exploration will just see the forest that is spanned by its queries. In other words, in the case of regular graphs, the only meaningful information that an exploration of the graph can obtain arises from encountering a simple cycle in the graph.

Hence, the core of the proof is the identification of a class of d-regular n-vertex graphs $\mathcal{G}_{d,n}$ such that, for any graph $G \in \mathcal{G}_{d,n}$, a $o(n^{1/2})$-step exploration of a random isomorphic copy of G is unlikely to encounter a cycle. Such an identification was provided in [14, Sec. 3.2], and it is pivoted at a parameter denoted sc and defined next.

To motivate this definition, we note that, as long as the exploration procedure does not encounter a simple cycle or a collision between two connected components, it is "practically non-adaptive" in the sense that its queries can be described by a fixed set of directed trees. Indeed, a *surprising event* occurs when a query made in one tree (unexpectedly)[6] hits a vertex that was already visited before (either in the same tree or in a different tree). Indeed, there are two different types of surprising events: (1) closing a cycle within the current tree, and (2) colliding with a different tree. It is easy to see that, in a q-query exploration, an

[5]The latter claim follows by a straightforward counting argument. Recalling that the number of labeled d-regular n-vertex graphs [5, 6] is $N_d(n) \stackrel{\rm def}{=} \Theta((dn/e)^{dn/2}/(d!)^n)$, where the Θ-notation hides a dependence on d, we observe that the number of d-regular n-vertex graphs that are ϵ-close to being isomorphic to a fixed graph is at most $M_{d,\epsilon}(n) \stackrel{\rm def}{=} n! \cdot \binom{dn/2}{\epsilon\cdot dn/2} \cdot n^{\epsilon\cdot dn/2}$. Hence, $M_{d,\epsilon}(n)/N_d(n)$ is upper-bounded by $(d/n)^{dn/2} \cdot n^{(1+0.5\epsilon d)\cdot n} \cdot 2^{H_2(\epsilon)\dot{d}n/2}$, which is negligible when $d \geq 3$ and $\epsilon \leq 1/2d$.

[6]Here we exclude the case that v was reached by making a query to vertex w, whereas a later query to vertex v returns w.

event of type (2) occurs with probability $O(q^2/n)$, and the focus of [14, Lem. 3.2.2] is on showing that the same bound holds for events of type (1).

Towards this end, it suffices to upper-bound the probability that a "blind" exploration of the graph yields some simple cycle, which in turn reduces to upper-bounding the probability that the next exploration-step closes a simple cycle. Lastly, as observed in [14, Sec. 3.2], the latter probability can be upper-bounded in terms of the probability that a *non-backtracking* random walk closes a simple cycle. A non-backtracking random walk is a walk that at each step chooses uniformly at random one of the neighbors of the current vertex other than the neighbor visited in the previous step [1]. Hence, following [14, Sec. 3.2], we consider the probability that a non-backtracking random walk consists of a simple cycle.

Definition 2.3.1 (probability of forming a simple cycle [14, Def. 3.2.1]): *For a graph $G = ([n], E)$, the* probability of forming a simple cycle in an ℓ-step random walk*, denoted $\mathtt{sc}_\ell(G)$, is the probability that a non-backtracking random walk that starts at a uniformly distributed vertex s reaches s in its ℓ^{th} step after visiting $\ell - 1$ distinct vertices. The* probability of forming a simple cycle in G*, denoted $\mathtt{sc}(G)$, is $\max_{\ell\in[n]}\{\mathtt{sc}_\ell(G)\}$.*

Note that $\mathtt{sc}_\ell(G) = 0$ if ℓ is either smaller than the girth of G or larger than n. Furthermore, $\mathtt{sc}_{\Omega(\log n)}(G)) = (1 \pm o(1))/n$ if G is a d-regular expander. More importantly, as shown in [14, Lem. 3.2.3], almost all d-regular n-vertex graphs G satisfy $\mathtt{sc}(G) = O(1/n)$. The following result is implicit in the proof of [14, Lem. 3.2.2].

Lemma 2.3.2 (the reduction): *For $d \geq 3$ and any d-regular graph $G = ([n], E)$, the probability that a surprising event occurs during a q-query exploration of a random isomorphic copy of G is upper-bounded by $O(q^2 \cdot \max(\mathtt{sc}(G), 1/n))$, where a surprising event is as defined above.*[7]

The bulk of the proof of [14, Lem. 3.2.2] is devoted to proving Lemma 2.3.2, where the notion of a surprising event is defined in the second paragraph of the proof and the foregoing claim is established one paragraph before its end. (The actual proof of [14, Lem. 3.2.2] goes-on and infers that the probability of "locating" an input vertex in a given canonical copy of G is upper-bounded analogously; but this is none of our business here.)

Using Lemma 2.3.2, we conclude that a $o(\sqrt{n})$-query exploration cannot distinguish random isomorphic copies of graphs that have linearly decreasing $\mathtt{sc}$-parameter; more generally

Corollary 2.3.3 (upper bounding the distinguishing gap): *For $d \geq 3$ and any two d-regular n-vertex graphs, G_1 and G_2, a q-query exploration of a random isomorphic copy of G_i cannot distinguish the case $i = 1$ from the case $i = 2$ with probability gap greater than $O(q^2 \cdot \max(\mathtt{sc}(G_1), \mathtt{sc}(G_2), 1/n))$, where the probability gap of an exploration is the quantity captured by the l.h.s of* Eq. (2.1).

Corollary 2.3.3 holds because the actual (label-invariant) information obtained by an exploration of a regular graph that encounters no surprising event is fully determined by the queries made during this exploration.

Combining Corollary 2.3.3 with [14, Lem. 3.2.3] (which asserts that almost all d-regular n-vertex graphs G satisfy $\mathtt{sc}(G) = O(1/n)$), we establish Theorem 2.1.2. More generally, we get

Corollary 2.3.4 (generalization of Theorem 2.1.2): *For $d \geq 3$, let H be a d-regular n-vertex graph, and let K be a d-regular n-vertex graph that is $\Omega(1)$-far from being isomorphic to H. Then, in the bounded-degree graph model with degree bound d, the query complexity of testing isomorphism to H is $\Omega(\min(\mathtt{sc}(H)^{-1/2}, \mathtt{sc}(K)^{-1/2}, n^{1/2}))$.*

[7]That is, a *surprising event* occurs when a query (unexpectedly) hits a vertex that was already visited before (either in the same tree or in a different tree). Recall that there are two different types of surprising events: (1) closing a cycle within the current tree, and (2) colliding with a different tree.

CHAPTER 3

TESTING GRAPH PROPERTIES WHEN THE DEGREE BOUND IS 2

3.1 Introduction

In this chapter we consider testing graph properties in the bounded-degree graph model, when the degree bound is 2. The bounded-degree model was introduced in [16], and is surveyed in [11, Chap. 9]. Even when considering only natural graph properties, the query complexity of testing in this model varies from being independent of the size of the graph to being linear in it (see [11, Sec. 9.6]). When allowing also unnatural properties, the complexity can be practically anything (see [15, 12]). In contrast to the richness of these results, which hold even for degree bound 3, we show that the situation with degree bound 2 is quite dull. Specifically, we show that, when the degree bound is two, every graph property can be tested within query complexity $\mathrm{poly}(1/\epsilon)$, where ϵ denotes the proximity parameter.

3.1.1 Background

The bounded-degree graph model refers to a fixed degree bound, denoted $d \geq 2$. An n-vertex graph $G = ([n], E)$, of maximum degree d, is represented in this model by a function $g : [n] \times [d] \to \{0, 1, ..., n\}$ such that $g(v, i) = u \in [n] \overset{\text{def}}{=} \{1, 2, ..., n\}$ if u is the i^{th} neighbor of v and $g(v, i) = 0$ if v has less than i neighbors. Hence, it is also adequate to refer to this model as the incidence function model. For simplicity, we assume here that the neighbors of vertex v appear in an arbitrary order in the sequence $g(v, 1), ..., g(v, \deg(v))$, where $\deg(v) \overset{\text{def}}{=} |\{i : g(v, i) \neq 0\}|$ is the degree of v. Also, we shall always assume that if $g(v, i) = u \in [n]$, then there exists $j \in [d]$ such that $g(u, j) = v$.

Distance between graphs is measured in terms of their aforementioned representation; that is, as the fraction of (the number of) different array entries over $d \cdot n$ (equiv., fraction of different edges over $dn/2$). We are interested in *graph properties*, which are sets of graphs that are closed under isomorphism; that is, Π is a graph property if for every graph $G = ([n], E)$ and every permutation π of $[n]$ it holds that $G \in \Pi$ if and only if $\pi(G) \in \Pi$, where $\pi(G) \overset{\text{def}}{=} ([n], \{\{\pi(u), \pi(v)\} : \{u, v\} \in E\})$. With these preliminaries in place, we now recall the meaning of property testing in this model.

Definition 3.1.1 (testing graph properties in the bounded-degree graph model): *For a fixed d, a* tester *for a graph property Π is a probabilistic oracle machine that, on input parameters n and ϵ, and access to* (the incidence function of) *an n-vertex graph $G = ([n], E)$ of maximum degree d, outputs a binary verdict that satisfies the following two conditions.*

1. *If $G \in \Pi$, then the tester accepts with probability at least* $2/3$.

2. *If G is ϵ-far from Π, then the tester accepts with probability at most* $1/3$*, where G is* ϵ-far *from Π if for every n-vertex graph $G' = ([n], E') \in \Pi$ of maximum degree d it holds that the symmetric difference between E and E' has cardinality that is greater than $\epsilon \cdot dn/2$.* (Equivalently, we may say that G is ϵ-far from G' if for every $g : [n] \times [d] \to \{0, 1, ..., n\}$ and $g' : [n] \times [d] \to \{0, 1, ..., n\}$ that represent G and G', respectively, it holds that $|\{(v, i) : g(v, i) \neq g'(v, i)\}| > \epsilon \cdot dn$.)[1]

If the tester accepts every graph in Π with probability 1, then we say that it has one-sided error*; otherwise, we say that it has* two-sided error*.*

The query complexity of a tester is the number of queries it makes to any n-vertex graph, as a function of the parameters n and ϵ.

3.1.2 Our results

Our main result (i.e., Theorem 3.2.4) is that, *when the degree bound equals 2, any graph property can be tested using* $\text{poly}(1/\epsilon)$ *queries* (in the bounded-degree graph model). The proof of this result is based on two simple observations:

1. A graph of maximum degree two consists of a collection of paths and cycles.

2. Each m-vertex graph that consists of paths and cycles of length at least ℓ is $1/\ell$-close to a single m-vertex path (resp., cycle).

Combining these two observations, when given oracle access to an n-vertex graph (of maximum degree 2), we can "approximately learn" the graph (up to isomorphism) by approximating the number of i-vertex cycles and paths, for each $i \in [O(1/\epsilon)]$. Given this approximation of the graph, one can readily determine its distance to any graph property. Likewise, one can determine whether two graphs are isomorphic (or far from being so).

Organization. In Section 3.2 we present a relatively straightforward implementation of the foregoing ideas, which yields testers of query complexity $O(1/\epsilon^4)$. A more sophisticated implementation, which yields an almost optimal bound of $\widetilde{O}(1/\epsilon^2)$, is presented in Section 3.3.

3.2 Main results

Starting from the foregoing two observations, we associate graphs with the numbers of vertices in each type of connected component, where the main types correspond to paths and cycles of specific small lengths (e.g., 4-vertex paths and 7-vertex cycles). In addition, we also consider isolated vertices and connected components of large size. Approximating the number of vertices that reside in connected components of each of these types is done by sampling vertices and exploring the connected component in which they reside while stopping the exploration when detecting that the connected component is larger than some designated bound (denoted ℓ).

Specifically, for $\ell = O(1/\epsilon)$, we consider 2ℓ types of connected components. The types correspond to possible lengths of paths and cycles, where we distinguish each of the lengths up to ℓ (i.e., $(i + 1)$-vertex paths and $(i + 2)$-vertex cycles for $i \in [\ell - 1]$) from isolated vertices (viewed as length-zero paths) and from paths and cycles of length greater than ℓ. Specifically,

- Type 1 correspond to isolated vertices;

- For $i \in [\ell-1]$ Type $2i$ corresponds to $(i+1)$-vertex paths, whereas Type $2i+1$ corresponds $(i+2)$-vertex cycles; and

[1]We say that G is ϵ-close to G' if G is not ϵ-far from G'; that is, $|\{(v, i) : g(v, i) \neq g'(v, i)\}| \leq \epsilon \cdot dn$.

- Type 2ℓ corresponds to either paths or cycles of length greater than in the other types (i.e., paths having more than ℓ vertices or cycles having more than $\ell+1$ vertices).

Note that a connected component with more than $\ell+1$ vertices can be detected by making at most $2\cdot(\ell+1)$ queries, and ditto for detecting an $(\ell+1)$-vertex path. For a graph $G=([n],E)$, we let g_i denote the fraction of vertices that reside in connected components of Type i, where $i\in[2\ell]$, and call the sequence of g_i's the ℓ-profile of G.

Algorithm 3.2.1 (main algorithm): *On input $n\in\mathbb{N}$ and $\epsilon>0$, given oracle access to a graph $G=([n],E)$, letting $\ell=O(1/\epsilon)$, we sample $O(\ell/\epsilon^2)$ vertices, and determine for each sampled vertex the type of connected component in which it resides, where the type of each sample vertex is determined by making at most $2\cdot(\ell+1)$ queries* (per each sampled vertex)*. We output $(\widehat{g}_1,....,\widehat{g}_{2\ell})$, such that $\widehat{g}_i$ denotes the fraction of sampled vertices that reside in connected components of Type i.*

The query (and time) complexity of this algorithm is $O(\ell^2/\epsilon^2)=O(1/\epsilon^4)$. Recall that a distribution over $[m]$ can be learned up to a total variation distance of ϵ by using $O(m/\epsilon^2)$ samples (see, e.g., [11, Exer. 11.4]). Letting the g_i's denote the actual ℓ-profile of the graph G, it follows that, *with* (say) *probability at least* 0.9*, it holds that*

$$\sum_{i\in[2\ell]}|\widehat{g}_i-g_i|<\epsilon.$$

Note that the $\widehat{g}_i$'s do not necessarily fit the profile of any n-vertex graph (e.g., they may not even be integer multiples of $1/n$). However, we shall consider and only make statements about the set of graphs that have an ℓ-profile that is close to $(\widehat{g}_1,....,\widehat{g}_{2\ell})$. Specifically, we prove that if the ℓ-profiles of two graphs are at distance at most ϵ, then they are $O(\epsilon)$-close to being isomorphic.

Lemma 3.2.2 (profiles versus isomorphism): *For $\epsilon>0$ and $\ell\geq 4/\epsilon$, let $(g_1,...,g_{2\ell})$ be the ℓ-profile of $G=([n],E)$ and $(g'_1,...,g'_{2\ell})$ be the ℓ-profile of $G'=([n],E')$. If $\sum_{i\in[2\ell]}|g_i-g'_i|\leq\epsilon$, then G is 2ϵ-close to being isomorphic to G'.*

Proof: We shall show that both G and G' are each ϵ-close to a graph with ℓ-profile $(g''_1,...,g''_{2\ell})$ such that $g''_i=\min(g_i,g'_i)$ for every $i\in[2\ell-1]$ (and $g''_{2\ell}=1-\sum_{i\in[2\ell-1]}g''_i$). Furthermore, this graph will contain a single connected component of Type 2ℓ, which will be a cycle. Although these two graphs (with ℓ-profile $(g''_1,...,g''_{2\ell})$) may be different, they are isomorphic to one another. Hence, we focus on proving that G is ϵ-close to a graph G'' that satisfies the foregoing conditions (and the agrument for G' is analogous).

We first observe that

$$\sum_{i\in[2\ell-1]}|g_i-\min(g_i,g'_i)| \;\leq\; \sum_{i\in[2\ell]:g_i>g'_i}(g_i-g'_i) \;=\; \frac{1}{2}\cdot\sum_{i\in[2\ell]}|g_i-g'_i|,$$

where equality holds if $g_{2\ell}\leq g'_{2\ell}$. Hence, $\sum_{i\in[2\ell-1]}|g_i-g''_i|\leq 0.5\cdot\epsilon$.

Let V_i denote the set of vertices of G that reside in connected component of Type i, and note that $|V_i|=g_i\cdot n$. For each $i\in[2\ell-1]$, we let V''_i be an adequate $g''_i\cdot n$-vertex subset of V_i, and let $V''_{2\ell}=V_{2\ell}\cup\bigcup_{i\in[2\ell-1]}(V_i\setminus V''_i)$. Specifically, for each $i\in[2\ell-1]$, the vertices in V''_i are complete connected components of G; that is, the movement does not split connected components of G. (This is the case because both $g_i\cdot n$ and $g''_i\cdot n$ are multiples of the number of vertices in each connected componet of Type i.)

Indeed, we aim at making V''_i be the set of vertices that reside in connected component of Type i in G''. Towards this end, we only need to modify the adjacencies of the vertices in $V''_{2\ell}$. The key observation is that the subgraph (of G) induced by $V''_{2\ell}$ can be made a single cycle by modifying the adjacencies of at most two vertices in each connected component in the subgraph (of G) induced by $V_{2\ell}\cup\bigcup_{i\in[2\ell-1]}(V_i\setminus V''_i)$. Recalling that connected components in $V_{2\ell}$ have size greater than ℓ, it follows that we only need to modify the adjacencies of $\frac{2}{\ell}\cdot|V_{2\ell}|+\sum_{i\in[2\ell-1]}(g_i-g''_i)\cdot n$ vertices. Using $|V_{2\ell}|\leq n$, $\ell\geq 4/\epsilon$ and $\sum_{i\in[2\ell-1]}|g_i-g''_i|\leq 0.5\cdot\epsilon$, the claim follows. ■

Applications. Using Algorithm 3.2.1 and Lemma 3.2.2, we derive the testing results stated in the introduction. We start with the problem of testing isomorphism between a pair of input graphs, which requires a straightforward extension of the testing model.

Theorem 3.2.3 (testing isomorphism): *In the bounded-degree graph model with degree bound 2, testing isomorphism between two input graphs can be done in time $O(1/\epsilon^4)$.*

Proof: The tester consists of approximating the profiles of both input graphs by invoking Algorithm 3.2.1, with proximity parameter $\epsilon/8$, and accepting if and only if the difference between the two estimated profiles is at most $\epsilon/4$.

If the graphs are isomorphic, then, with probability at least 0.8, the estimated profiles are at distance at most $2 \cdot \epsilon/8$ from one another. On the other hand, if the graphs are ϵ-far from being isomorphic, then (by Lemma 3.2.2) their profiles are at distance greater than $\epsilon/2$ from one another. In that case, with probability at least 0.8, their estimated profiles are at distance greater than $\frac{\epsilon}{2} - 2 \cdot \frac{\epsilon}{8} = \epsilon/4$ from one another. ■

Theorem 3.2.4 (testing any graph property): *Let Π be an arbitrary graph property. Then, in the bounded-degree graph model with degree bound 2, the property Π can be tested using $O(1/\epsilon^4)$ queries.*

Proof: The tester consists of approximating the profile of the input graph by invoking Algorithm 3.2.1, with proximity parameter $\epsilon/4$, and accepting if and only if the difference between the estimated profile and the profile of some graph in the property is at most $\epsilon/4$.

Evidently, if the input graph is in Π, then its profile fits a profile of a graph in Π, and the tester will accept with probability at least 0.9. On the other hand, if the graph is ϵ-far from any graph in the property, then (by Lemma 3.2.2) its profile is at distance greater than $\epsilon/2$ from the profile of any graph in the property. In that case, with probability at least 0.9, its estimated profile is at distance greater than $\frac{\epsilon}{2} - \frac{\epsilon}{4} = \epsilon/4$ from any such profile. ■

3.3 Limitations and Improvements

The proofs of both Theorems 3.2.3 and 3.2.4 yield two-sided error testers. As noted in [13, Thm. 2.5], two-sided error is inherent to testing isomorphism with sub-linear query complexity in this model (even with degree bound 2). This holds even when testing isomorphism to a fixed n-vertex graph consisting of $n/6$ isolated triangles and $n/6$ length 2 paths.[2] Hence, two-sided error in also inherent to Theorem 3.2.4.

The aforementioned graph also demonstrates that both testing tasks require $\Omega(1/\epsilon^2)$ queries.[3] Furthermore, it seems that $\Omega(1/\epsilon^4)$ queries are needed in order to obtain an approximated profile in the sense obtained by Algorithm 3.2.1 (i.e., $(\widehat{g}_1, ..., \widehat{g}_{2\ell})$ such that $\sum_{i\in[2\ell]} |\widehat{g}_i - g_i| \leq \epsilon$, where $(g_1, ..., g_{2\ell})$ is the ℓ-profile of the input graph and $\ell = \Theta(1/\epsilon)$).[4] However, such an approximated profile is not required in order to perform the foregoing testing tasks. As argued next, it suffices to provide less accurate approximations of the number of vertices that belong to larger connected components.

The key observation is that the cost of moving vertices from connected components of one type to components of a different type is inversely proportional to the size of the smaller connected components (e.g., modifying m paths that are each of length ℓ' into a cycle of length $m\ell'$ can be achieved by $2m$ adjacency modification). On the other hand, the cost of determining the type of a connected component is upper bounded by its size, and so we can afford to provide a better approximation for the number of vertices in them.

[2] Actually, we can establish the claim also with respect to degree bound 1, by using a fixed n-vertex graph consisting of $n/2$ isolated vertices and $n/4$ isolated edges.

[3] Consider the task of testing whether an input graph is isomorphic to the aforementioned fixed graph, denoted F. Then, $\Omega(1/\epsilon^2)$ queries are required to distinguish a random isomorphic copy of F from a random n-vertex graph that consists of $(1+3\epsilon)n/6$ isolated triangles and $(1-3\epsilon)n/6$ length 2 paths.

[4] Specifically, learning an unknown distribution over $[m]$ up to a total variation distance of ϵ requires $\Omega(m/\epsilon^2)$ samples, whereas determining the type of a connected component (wrt types in $[2\ell]$) requires $\Omega(\ell)$ queries.

The simpler implementation of the foregoing idea uses a *single threshold* (of $\sqrt{\ell}$), and distinguishes between small connected components (which have size smaller than this threshold) and larger ones. A more sophisticated implementation clusters the connected components according to their approximated size, and uses $O(\log \ell)$ such *clusters* (rather than two). These two implementations are presented in the following two subsections.

3.3.1 The single size-threshold algorithm

We focus on the design and analysis of a more efficient algorithm that replaces Algorithm 3.2.1. As before, we set $\ell = O(1/\epsilon)$. We use $t = \sqrt{\ell}$ as the threshold distinguishing the first $2t-1$ types from the remaining $2\ell - 2t + 1$ types. We approximate the first g_i's by $\widehat{g}_i$'s such that $\sum_{i\in[2t-1]} |\widehat{g}_i - g_i| < \epsilon$, using a sample of size $O(t/\epsilon^2)$, at the cost of making $O(t)$ queries per sampled vertex. In contrast, the remaining g_i's is approximated by $\widehat{g}_i$'s such that $\sum_{i\in[2t,2\ell]} |\widehat{g}_i - g_i| < t\cdot\epsilon$, using a sample of size $O(\ell/(t\cdot\epsilon)^2)$, at the cost of making $O(\ell)$ queries per sampled vertex. Specifically, the algorithm proceeds as follows.

1. We sample $O(t/\epsilon^2)$ vertices and determine $\widehat{g}_1, ..., \widehat{g}_{2t-1}$ by making at most $2\cdot(t+1)$ queries per each sampled vertex. As before (i.e., in Algorithm 3.2.1), $\widehat{g}_i$ denotes the fraction of sampled vertices that reside in connected components of Type i.

2. Letting $\eta = t\cdot\epsilon$, we sample $O(\ell/\eta^2)$ vertices and determine $\widehat{g}_{2t}, ..., \widehat{g}_{2\ell}$ by making at most $2\cdot(\ell+1)$ queries (per each sampled vertex).

The query (and time) complexity of this algorithm is $O(t^2/\epsilon^2) + O(\ell^2/\eta^2) = O(1/\epsilon^3)$, since $t^2 = \ell = O(1/\epsilon)$. Letting the g_i's denote the actual profile of the input graph G, and using [11, Exer. 11.4], it follows that, *with* (say) *probability at least* 0.9*, it holds that* $\sum_{i\in[2t-1]} |\widehat{g}_i - g_i| < \epsilon$ *and* $\sum_{i\in[2t,2\ell]} |\widehat{g}_i - g_i| < \eta$. Again, we shall consider the set of graphs that have an ℓ-profile that is close to $(\widehat{g}_1, ..., \widehat{g}_{2\ell})$, but the notion of "being close" is adapted to fit the approximation provided by the foregoing algorithm.

Lemma 3.3.1 (a variant on Lemma 3.2.2): *For $\epsilon > 0$, $\ell = 2/\epsilon$ and $t = \sqrt{\ell}$, let $(g_1, ..., g_{2\ell})$ be the ℓ-profile of $G = ([n], E)$ and $(g'_1, ..., g'_{2\ell})$ be the ℓ-profile of $G' = ([n], E')$. If $\sum_{i\in[2t-1]} |g_i - g'_i| \le \epsilon$, $\sum_{i\in[2t,2\ell]} |g_i - g'_i| \le \eta \stackrel{\rm def}{=} t\cdot\epsilon$, then G is 8ϵ-close to being isomorphic to G'.*

Using Lemma 3.3.1, we obtain improvements over the complexity bounds stated in Theorems 3.2.3 and 3.2.4. Specifically, the bound is improved from $O(1/\epsilon^4)$ to $O(1/\epsilon^3)$.

Proof: Intuitively, when moving vertices between two different types that correspond to connected components of size greater than t it suffices to modify the adjacencies of a $2/t$ fraction of the vertices. Hence, the second sum (i.e., $\sum_{i\in[2t,2\ell]} |g_i - g'_i|$), which is smaller than η, contributes only $O(\eta/t) = O(\epsilon)$ units to the difference between G and G'. The contribution of the first sum (i.e., $\sum_{i\in[2t-1]} |g_i - g'_i|$), remains $O(\epsilon)$ as before. Details follow.

As in the proof of Lemma 3.2.2, we shall actually prove that each of the two graphs (i.e., G and G') is 4ϵ-close to being isomorphic to a graph G'' that has the ℓ-profile $(g''_1, ..., g''_{2\ell})$ such that $g''_i = \min(g_i, g'_i)$ for every $i \in [2\ell - 1]$ (and $g''_{2\ell} = 1 - \sum_{i\in[2\ell-1]} g''_i$). Furthermore, G'' will contain a single connected component of Type 2ℓ, which will be a cycle. The construction of G'' is identical to the one in the proof of Lemma 3.2.2, and all that changes is the analysis of the number of vertices whose adjacency is modified.

Recall that we upper-bounded this number by two vertices per each connected component in the subgraph (of G) induced by $V_{2\ell} \cup \bigcup_{i\in[2\ell-1]} (V_i \setminus V''_i)$. Here, we use the fact that each of the connected components in the subgraph induced by $\bigcup_{i\in[2t,2\ell-1]} (V_i \setminus V''_i)$ have size greater than t. It follows that we only need to modify the adjacencies of at most

$$\frac{2}{\ell}\cdot|V_{2\ell}| + \sum_{i\in[2t-1]} (g_i - g''_i)\cdot n + \frac{2}{t}\cdot\sum_{i\in[2t,2\ell-1]} (g_i - g''_i)\cdot n$$

vertices. Using $\sum_{i\in[2t-1]} |g_i - g''_i| \le \epsilon$ and $\sum_{i\in[2t,2\ell-1]} |g_i - g''_i| \le t\cdot\epsilon$ (as well as $|V_{2\ell}| \le n$ and $\ell \ge 2/\epsilon$), the claim follows. ■

3.3.2 The size-clustering algorithm

As before, we set $\ell = O(1/\epsilon)$, but here ℓ is a power of 2. We partition $[2\ell-1]$ into $\ell' = \log_2(2\ell)$ intervals, denoted $I_1, ..., I_{\ell'}$, such that $I_k = [2^{k-1}, 2^k - 1]$. Our main goal will be to approximate the g_i's by $\widehat{g}_i$'s such that $\sum_{i\in I_k} |\widehat{g}_i - g_i| < \epsilon_k \stackrel{\rm def}{=} 2^{k-1}\cdot\epsilon$ for every $k \in [\ell']$. (Note that $|\widehat{g}_{2\ell} - g_{2\ell}| < 2^{\ell'-1}\cdot\epsilon$ always holds, since we shall use $\ell > 1/\epsilon$.) For each $k \in [\ell']$, we approximate the $(g_i)_{i\in I_k}$'s by sampling $O(t\cdot|I_k|/\epsilon_k^2)$ vertices, making at most $2\cdot((2^k-1)+1) = 2^{k+1}$ queries per each sampled vertex, and determining the $\widehat{g}_i$'s for each $i \in I_k$; specifically, we use the following algorithm.

Algorithm 3.3.2 (main algorithm, revisited): *On input $n \in \mathbb{N}$ and $\epsilon > 0$, given oracle access to a graph $G = ([n], E)$, we set $\ell = 2^{\ell'-1} = O(1/\epsilon)$ and $t = \log_2(10\ell')$. For each $k \in [\ell']$, recalling that $I_k = [2^{k-1}, 2^k-1]$ and $\epsilon_k = 2^{k-1}\cdot\epsilon$, we proceed as follows.*

- *We sample $O(t\cdot|I_k|/\epsilon_k^2)$ vertices. For each sampled vertex, making at most 2^{k+1} queries, we determine whether it resides in a connected component with type in I_k, and if so the type of this connected component.*
- *For each $i \in I_k$, we let $\widehat{g}_i$ denote the fraction of sampled vertices that reside in connected components of Type i.*

We output $(\widehat{g}_1, ..., \widehat{g}_{2\ell})$, where $\widehat{g}_{2\ell} \leftarrow 1 - \sum_{i\in[2\ell-1]} \widehat{g}_i$.

Hence, for each k, with probability at least $1-2^{-t}$, it holds that $\sum_{i\in I_k} |\widehat{g}_i - g_i| < \epsilon_k$, and using $t = \log_2(10\ell')$ implies that all ℓ' inequalities hold with probability at least 0.9. The query (and time) complexity of Algorithm 3.3.2 is

$$\sum_{k\in[\ell']} O(t\cdot|I_k|/\epsilon_k^2)\cdot O(2^k) \;=\; \sum_{k\in[\ell']} O(t\cdot 2^{2k}/(2^{k-1}\cdot\epsilon)^2) \;=\; O(t\cdot\ell'/\epsilon^2) \;=\; \widetilde{O}(\ell')/\epsilon^2.$$

Again, we shall consider the set of graphs that have an ℓ-profile that is close to $(\widehat{g}_1, ..., \widehat{g}_{2\ell})$, but the notion of "being close" is adapted to fit the approximation provided by Algorithm 3.3.2.

Lemma 3.3.3 (another variant on Lemma 3.2.2): *For $\epsilon > 0$, $\ell' = \lceil\log_2(4/\epsilon)\rceil$ and $\ell = 2^{\ell'-1}$, let $I_k = [2^{k-1}, 2^k-1]$ and $\epsilon_k = 2^{k-1}\cdot\epsilon$. If $\sum_{i\in I_k} |g_i - g'_i| \le \epsilon_k$ for every $k \in [\ell']$, where $\overline{g} = (g_1, ..., g_{2\ell})$ is the ℓ-profile of $G = ([n], E)$ and $\overline{g}' = (g'_1, ..., g'_{2\ell})$ is the ℓ-profile of $G' = ([n], E')$, then G is $O(\ell'\cdot\epsilon)$-close to being isomorphic to G'.*

Proof: We generalized the proof of Lemma 3.3.1, using again the same construction of an intermediate graph G'' as in the proof of Lemma 3.2.2, and adapting the analysis to the current setting. That is, we shall show that each of the two graphs (i.e., G and G') is $O(\ell'\cdot\epsilon)$-close to being isomorphic to a graph G'' that has the ℓ-profile $(g''_1, ..., g''_{2\ell})$ such that $g''_i = \min(g_i, g'_i)$ for every $i \in [2\ell-1]$ (and $g''_{2\ell} = 1 - \sum_{i\in[2\ell-1]} g''_i$). (Again, G'' will contain a single connected component of Type 2ℓ, which will be a cycle.)

Recall that we upper-bounded the number of vertices whose adjacencies is modified by two vertices per each connected component in the subgraph (of G) induced by $V_{2\ell} \cup \bigcup_{i\in[2\ell-1]}(V_i \setminus V''_i)$. Here we use the fact that the connected components in the subgraph induced by $\bigcup_{i\in I_k}(V_i \setminus V''_i)$ have size greater than 2^{k-2}. It follows that we only need to modify the adjacencies of at most

$$\frac{2}{\ell}\cdot|V_{2\ell}| + \sum_{k\in[\ell']} \frac{2}{2^{k-2}} \cdot \sum_{i\in I_k} (g_i - g''_i)\cdot n$$

vertices. Using $\sum_{i\in I_k} |g_i - g''_i| \le \epsilon_k = 2^{k-1}\cdot\epsilon$ (as well as $|V_{2\ell}| \le n$ and $\ell \ge 2/\epsilon$), we derive an upper bound of $\epsilon\cdot n + \sum_{k\in[\ell']} 4\epsilon\cdot n$, and the claim follows. ■

Theorem 3.3.4 (testing isomorphism, revisited): *In the bounded-degree graph model with degree bound 2, testing isomorphism between two input graphs can be done in time $\widetilde{O}(1/\epsilon^2)$.*

Proof: The tester consists of approximating the profiles of both input graphs by invoking Algorithm 3.3.2, with proximity parameter $\epsilon' \stackrel{\rm def}{=} \epsilon/O(\log(1/\epsilon))$, and accepting if and only if for every k it holds that the (norm-1) difference between the approximated frequencies that correspond to types in I_k is at most $2^k \cdot \epsilon'$. Note that the time (and query) complexity of this tester is $\widetilde{O}(\log(1/\epsilon')) \cdot (1/\epsilon')^2 = \widetilde{O}(\log^3(1/\epsilon)) \cdot (1/\epsilon)^2$.

If the graphs are isomorphic, then, with probability at least 0.8, the estimated frequencies for each k are at distance at most $2 \cdot 2^{k-1} \cdot \epsilon'$, and the tester accepts. On the other hand, if the graphs are ϵ-far from being isomorphic, then (by Lemma 3.3.3), for some k, the difference between the actual frequencies that corresponding to types in I_k is greater than $4 \cdot 2^{k-1} \cdot \epsilon'$. In that case, the corresponding estimates are at distance greater than $4 \cdot 2^{k-1} \cdot \epsilon' - 2 \cdot 2^{k-1} \cdot \epsilon'$ (with probability at least 0.8). ■

Theorem 3.3.5 (testing any graph property, revisited): *Let Π be an arbitrary graph property. Then, in the bounded-degree graph model with degree bound 2, the property Π can be tested using $\widetilde{O}(1/\epsilon^2)$ queries.*

Proof: The tester consists of approximating the profile of the input graph G by invoking Algorithm 3.3.2, with proximity parameter $\epsilon' = \epsilon/O(\log(1/\epsilon))$, and accepting if and only if there exists a graph G' in Π such that for every k it holds that the (norm-1) difference between the frequencies (of G' and G) that correspond to types in I_k is at most $2^{k-1} \cdot \epsilon'$.

Evidently, if the input graph is in Π, then its profile fits a profile of a graph in Π, and the tester will accept with probability at least 0.9. On the other hand, if the graph G is ϵ-far from any graph G' in the property, then (by Lemma 3.3.3), for some k, the difference between the actual frequencies (of G and G') that corresponding to types in I_k is greater than $2 \cdot 2^{k-1} \cdot \epsilon'$. In that case, the corresponding estimates of the frequencies of G are at distance greater than $2 \cdot 2^{k-1} \cdot \epsilon' - 2^{k-1} \cdot \epsilon'$ (with probability at least 0.9, regardless of G'). ■

CHAPTER 4

TESTING GROUP PROPERTIES

4.1 Introduction

The area of property testing first emerged in the context of testing algebraic properties such as group homomorphism (e.g., linearity) [4] and low-degree polynomials [29]. Hence, it is surprising that the problem of testing whether a matrix represents a group operation was first considered only a decade later, in [7], which is focused on a generalization (of property testing) called *spot checking.*[1]

Definition 4.1.1 (testing the property of being a group): *A* tester for being a group *is a probabilistic oracle machine that, on input parameters $n \in \mathbb{N}$ and $\epsilon > 0$, and oracle access to a function $f : [n] \times [n] \to [n]$ distinguishes between the following two cases.*[2]

f ***represents a group operation:*** *The set $[n] \overset{\text{def}}{=} \{1, 2, ..., n\}$ with the binary operation f is a group.*

f ***is ϵ-far from representing a group operation:*** *For every $g : [n] \times [n] \to [n]$ such that $[n]$ with g constitutes a group, it holds that f is ϵ-far from g; that is, $|\{(x, y) \in [n]^2 : f(x, y) \neq g(x, y)\}| > e \cdot n^2$.*

The tester is said to have one-sided error *if it always accepts any function that represents a group operation.*

Ergun *et al.* [7] presented a tester for the group property that has running time $\widetilde{O}(n^{3/2}/\epsilon)$. Specifically, in [7, Sec. 4.1], they claimed a tester of time complexity $\widetilde{O}(n/\epsilon)$ under the guarantee that the function f is cancellative (i.e., $f(x, y) = f(x, y')$ implies $y = y'$ and $f(x, y) = f(x', y)$ implies $x = x'$), whereas in [7, Sec. 4.3] they waived this hypothesis/guarantee and claimed a tester of time complexity $\widetilde{O}(n^{3/2}/\epsilon)$.[3] Our main result is

Theorem 4.1.2 (efficiently testing the property of being a group (see Corollary 4.4.3)): *There exists a $\widetilde{O}(n/\epsilon)$-time tester for the property of being a group. Furthermore, the tester has one-sided error.*

The tester consists of natural tests of associativity and cancellativity. Actually, we use specific forms of these intuitive tests, whereas it is not clear whether different natural forms of the same intuitive tests will do.

[1]This generalization contains, as special cases, notions such as property testing, PCPs of proximity (see [3]), and interactive proofs of proximity (see [28]).

[2]That is, the tester accepts any function that represents groups with probability at least 2/3, and rejects any function that is far from representing any group with probability at least 2/3.

[3]We believe that there in a gap in the proof of Lemma 34 of [7, Sec. 4.1]: The second equality in the probabilistic claim seems to replace $\odot'$ by $\odot$ with no justification. This affect the correctness of both [7, Thm. 28] and [7, Thm. 49].

We comment that focusing on the query complexity while ignoring the time complexity allows for a much simpler tester (for being a group). In fact, a generic test that applies to any *property of groups* is quite straightforward. Towards presenting this result, we generalized Definition 4.1.1 as follows.

Definition 4.1.3 (testing properties of groups): *Let $\mathcal{G}$ be a set of finite groups that is closed under isomorphism; that is, for every group G in $\mathcal{G}$, all groups that are isomorphic to G are in $\mathcal{G}$.*[4] *A* tester for being in $\mathcal{G}$ *is a probabilistic oracle machine that, on input parameters $n \in \mathbb{N}$ and $\epsilon > 0$, and oracle access to a function $f : [n] \times [n] \to [n]$ distinguishes between the following two cases.*

f ***represents the operation of a group in*** $\mathcal{G}$**:** *The set $[n] \stackrel{\text{def}}{=} \{1, 2, ..., n\}$ with the binary operation f is a group in $\mathcal{G}$.*

f ***is*** ϵ***-far from representing the operation of a group in*** $\mathcal{G}$**:** *For every $g : [n] \times [n] \to [n]$ such that $[n]$ with g constitutes a group in $\mathcal{G}$, it holds that f is ϵ-far from g; that is, $|\{(x, y) \in [n]^2 : f(x, y) \neq g(x, y)\}| > e \cdot n^2$.*

We may call $\mathcal{G}$ a property of groups. Using the fact that the number of groups over $[n]$ is $\exp(\widetilde{O}(n))$, we have

Theorem 4.1.4 (testing any property of groups (see Theorem 4.2.2)): *For any set $\mathcal{G}$ of finite groups that is closed under isomorphism, there exists a $\widetilde{O}(n/\epsilon)$-query tester for $\mathcal{G}$. Furthermore, the tester has one-sided error.*

Note that, unlike Theorem 4.1.2, which provides a computationally efficient tester, Theorem 4.1.4 does not bound the running time of the tester. Our proof of Theorem 4.1.4 uses a tester that runs in time that is exponential in its query complexity. Recall that Theorem 4.1.2 (as well as Theorem 4.1.5) does bound the time-complexity by $\widetilde{O}(n/\epsilon)$, but it only refers to the set of all groups (resp., all Abelian groups).

Yet another tester for the property of being a group is presented in Theorem 4.3.2. This tester uses $\widetilde{O}(n/\epsilon)$ queries but runs in time $\widetilde{O}(n^2)$ and has two-sided error. However, its proof is much simpler than the proof of Theorem 4.1.2. We mention that all three aforementioned testers can be augmented to test Abelian groups. In particular, we prove

Theorem 4.1.5 (efficiently testing the property of being an Abelian group (see Theorem 4.4.4)): *There exists a $\widetilde{O}(n/\epsilon)$-time tester for the property of being an Abelian group. Furthermore, the tester has one-sided error.*

(Note that $\widetilde{O}(n/\epsilon) = \widetilde{O}(n)/\epsilon$, because we may assume (w.l.o.g.) that $\epsilon \geq 1/n^2$.)

4.1.1 Related work

Following the work of Ergun *et al.* [7], studies of testing properties of groups appeared in two works [10, 21]. The first work (i.e., [10]) studies a related model in which the distance measure allows for an n-element group to be close to an $(n + 1)$-element group, whereas the second work studies both the model of [10] and the model that we use (i.e., Definition 4.1.3). Both works focus on testing properties of groups such as being Abelian and being cyclic. The results of [21] that refer to Definition 4.1.3 are (1) for every fixed $k \geq 2$, a $\Omega(n^{\frac{1}{6} - \frac{4}{6(3k+1)}})$ lower bound on the query complexity of testing whether an Abelian group is generated by k elements, and (2) a polylogarithmic-time tester for cyclic groups.

We stress that testers are oracle machines that are given access to the main input, which is a function from $[n] \times [n]$ to $[n]$. Hence, bounds on their running time postulates that each query to the oracle is answered in unit time. Note that this model is essentially equivalent to the standard RAM model, except that the testers may afford not reading the entire input. In contrast, exact decision procedures as presented in [27, 8] do read the entire input, and have running-time $O(n^2)$, which is optimal for exact decision.

[4]Two groups are called isomorphic if there exists a bijective group homomorphism between them.

4.1.2 Future directions

Theorems 4.1.2 and 4.1.5 improve over the previously known results regarding testing the property of being a group and being an Abelian group, respectively. However, it is unclear whether the complexities of these new testers are the best possible. In particular, it is possible that the complexity of testing Abelian groups is lower than the complexity of testing general groups.

We mention that a logarithmic lower bound on the query complexity of testing any group property that contains cyclic groups is not hard to prove (see Section 4.5), but this leaves a large gap towards the upper bound known for testing general (resp., Abelian) groups.

Theorem 4.1.6 (a lower bound on testing properties of groups (see Theorem 4.5.1)): *Let $\mathcal{G}$ be a set of finite groups that is closed under isomorphism and contains all cyclic groups of prime order. Then, testing $\mathcal{G}$ requires $\Omega(\log n)$ queries.*

In particular, it follows that testing cyclic groups requires at least a logarithmic (n the group size) number of queries. Recall that the upper bound of [21, Thm. 3] asserts that polylogarithmic complexity suffices for this task (i.e., for testing cyclic groups).

In general, determining the complexity of testing other group properties opens an interesting direction of research. Indeed, the aforementioned results of [21] (regarding Abelian groups generated by k elements) fit this direction, but leave much to be understood.

4.1.3 Organization

Our three testers of the group property are presented in three sections that are independent of one another. Our main result (i.e., Theorem 4.1.2) is presented in Section 4.4 (which contains also a proof of Theorem 4.1.5). The other sections are much easier, but reading them is not a useful warm-up towards reading Section 4.4. Section 4.2 presents a generic tester (which establishes Theorem 4.1.4), whereas Section 4.3 presents an alternative tester that makes $\widetilde{O}(n/\epsilon)$ queries but runs in time $\widetilde{O}(n^2)$ and has two-sided error.

The proof of Theorem 4.1.6, presented in Section 4.5, can be read independently of all other sections.

4.2 A generic result

A generic tester for any property (equiv., set) of groups $\mathcal{G}$ is derived from a generic tester for any property (equiv., set) of functions. Indeed, let $\mathcal{F}$ be an arbitrary set of functions from D to R. Then, $\mathcal{F}$ can be tested within query complexity $O(\epsilon^{-1}\log|\mathcal{F}|)$.

Theorem 4.2.1 (a totally generic tester): *Testing whether $f : D \to R$ is in the set $\mathcal{F}$ can be done using $O(\epsilon^{-1}\log|\mathcal{F}|)$ queries, and $\widetilde{O}(|\mathcal{F}|/\epsilon)$ time. Furthermore, the tester has one-sided error.*

Proof: The tester just picks $s = O(\epsilon^{-1}\log|\mathcal{F}|)$ random points in $[n]$ and checks whether the value of f on them is consistent with some function in $\mathcal{F}$. (Indeed, the time complexity of this tester is $O(s \cdot |\mathcal{F}|)$.)

Clearly, this tester always accepts any $f \in \mathcal{F}$. On the other hand, if f is ϵ-far from $\mathcal{F}$, then for every $f' \in \mathcal{F}$ it holds that

$$\Pr_{a_1,\ldots,a_s \in D}[(\forall i \in [s])\, f(a_i) = f'(a_i)] < (1-\epsilon)^s.$$

Using a union bound (and noticing that $(1-\epsilon)^s = o(1/|\mathcal{F}|)$), it follows that the tester rejects f with probability at least 2/3. ■

Theorem 4.2.2 (generic tester of group properties (Theorem 4.1.4, restated)): *Let $\mathcal{G}_n$ be a set of n-element groups. Then, testing whether $f : [n] \times [n] \to [n]$ describes a group that is isomorphic to a group in $\mathcal{G}_n$ can be done using $\widetilde{O}(n)/\epsilon$ queries, and $\exp(\widetilde{O}(n))/\epsilon$ time. Furthermore, the tester has one-sided error.*

Proof: The key observation is that the number of non-isomorphic groups over $[n]$ is $\exp(O(\log n)^3)$; see [24, 22, 26]. Now, letting $\mathcal{F}$ be the set of all functions obtained by relabelling all $g \in \mathcal{G}_n$, the task of testing the group property $\mathcal{G}_n$ reduces to testing membership in $\mathcal{F}$, whereas $|\mathcal{F}| \leq n! \cdot |\mathcal{G}_n|$. Using Theorem 4.2.1, the claim follows. ■

4.3 Testing whether f represents a group, take 2

The down side of Theorem 4.2.2 is its prohibitively high computational complexity (i.e., $\exp(\widetilde{O}(n))$-time). Here, we significantly improve the computational complexity at the cost of allowing two-sided error probability, but we still do not reach sub-linear (in n^2) time.

The basic idea is any group can be generated by a small set of expanding generators; specifically, for any group of n elements, a random set of $O(\log n)$ elements can serve as a set of expanding generators [2], where expanding means that composing a random $O(\log n)$-long sequence of generators (and inverses) yields an almost uniformly distributed element. Hence, the group operation (i.e., $[n] \times [n] \to [n]$) is fully specified by the values assigned to pairs of the form $[n] \times S$, where S is a generating set (which includes the inverses of elements in S). Specifically, we construct the following succinct data structure that allows to perform group operations in logarithmic time.

Construction 4.3.1 (producing a succinct representation of an alleged group): *Given oracle access to $f : [n] \times [n] \to [n]$, we proceed as follow.*

1. Finding a neutral element. *This is done by picking an arbitrary $a \in [n]$, and finding the unique $e \in [n]$ that satisfies $f(a,e) = f(e,a) = a$.*

   If this or any of the next steps fails, then we halt while indicating failure.

2. Finding a set of expanding generators (along with their inverses). *We select uniformly at random a set S of $O(\log n)$ elements in $[n]$, and for each $s \in S$ find the unique $s' \in [n]$ such that $f(s,s') = f(s',s) = e$.*

   We redefine S to contain S as well as the foregoing inverses. We also record $I : S \to S$ such that $I(s)$ is the inverse of s.

3. Constructing the main table. *Querying f on $[n] \times S$, we construct $T : [n] \times S \to [n]$ such that $T(a,s) = f(a,s)$ for every $a \in [n]$ and $s \in S$.*

   (Actually, this information is already available to us (from the search for inverses of S performed in the previous step).)

4. Expressing all elements in terms of the generating set. *We generate $O(n \log n)$ random sequences over S, each of length $\ell = O(\log n)$, in hope of reaching all n elements of the group. The element generated by a sequence $(s_1, ..., s_\ell) \in S^\ell$ is obtained by repeated composition. Specifically, for each sampled sequence $(s_1, ..., s_\ell) \in S^\ell$, we compute the value $f(\cdots(f(f(s_1,s_2),s_3),...),s_\ell)$ by using $T(\cdots(T(T(s_1,s_2),s_3),...),s_\ell)$.*

   For each $x \in [n]$, we let $(s_1^x, ..., s_\ell^x)$ denote a sequence that is used to generate x. We define $E : [n] \times [\ell] \to S$ such that $E(x,i) = s_i^x$.

   Note that no queries are made at this step.

We output e, S and the tables I, T and E.

Overall, we made $O(n) + O(n \log n) + O(n \log n)$ queries and spent $\widetilde{O}(n)$ time. We stress that if f represents a group, then, with high probability, S is a set of expanding generators, and the sample of ℓ-sequences reaches all elements of the group. In this case, the construction returns the desired tables. In any case, if Construction 4.3.1 returns tables as stipulated, then we define the following function $f' : [n] \times [n] \to [n]$

$$f'(x,y) \overset{\text{def}}{=} T(\cdots(T(T(x,E(y,1)),E(y,2)),...),E(y,\ell)). \tag{4.1}$$

That is, starting with $v \leftarrow x$, for $i = 1, ..., \ell$, we let $v \leftarrow T(v, E(y, i))$, and finally let $f'(x, y) \leftarrow v$. Note that evaluating f' amounts to $O(\log n)$ accesses to the tables E and T. Hence, testing whether f represents a group (resp., a group with certain properties) reduces to invoking Construction 4.3.1, testing whether f' equals f, and checking that f' satisfies the group property (resp., properties).

Theorem 4.3.2 (testing the group property): *Testing whether $f : [n] \times [n] \to [n]$ represents a group can be done using $\widetilde{O}(n) + O(1/\epsilon)$ queries, and $\widetilde{O}(n^2) + O(\epsilon^{-1} \cdot \log n)$ time.*

Note that this tester has two-sided error probability, which is due to the (small) error probability of Construction 4.3.1.

Proof: As stated upfront, we first invoke Constuction 4.3.1 and reject if it failed. Otherwise, we obtain tables that allow us to compute f' in logarithmic time, while making no additional queries, where f' is as defined in Eq. (4.1). In this case, we check that f' represents a group, and make $O(1/\epsilon)$ additional queries (to f) in order to test that f equals f' (while rejecting (w.h.p.) when f is ϵ-far from f'). Specifically, checking that f' represents a group is done by explicitly constructing f' (according to Eq. (4.1)) and checking all group axioms (see details below). Testing equality between f and f' amounts to querying f at $O(1/\epsilon)$ random entries, and comparing these value to the corresponding values obtained from f'.

As for checking that f' satisfies all group axioms, a straightforward implementation allows checking the unique existence of a neutral element and the inverses in $O(n^2)$ time, whereas associativity is checked in $O(n^3)$ time. We improve upon the latter by observing that there is no need to check all triplets in $[n]$; it suffices to check triplets in which one of the elements is confined to S. Actually, this observation was made by Light in 1949 (see [27]), and we prove it next for sake of self-containment.

Claim: If f' satisfies $f'(x, f'(y, z)) = f'(f'(x, y), z)$ for all $x, y, z \in [n]$ such that $\{x, y, z\} \cap S \neq \emptyset$, then f' satisfies $f'(x, f'(y, z)) = f'(f'(x, y), z)$ for all $x, y, z \in [n]$.

Proof: Let $\overline{f}(s) \stackrel{\rm def}{=} s$ for every $s \in S$. Using induction on $t \geq 1$, we define $\overline{f} : S^t \to [n]$ such that $\overline{f}(s_1, s_2, ..., s_t) \stackrel{\rm def}{=} f'(s_1, \overline{f}(s_2, ..., s_t))$. We first prove that

$$f'(\overline{f}(s_1, ..., s_{t'}), \overline{f}(s_{t'+1}, ..., s_t)) = f'(\overline{f}(s_1, ..., s_{t''}), \overline{f}(s_{t''+1}, ..., s_t)) \qquad (4.2)$$

for every $t', t'' \in [t-1]$.
The special case of $t' = 1$ implies that $\overline{f}(s_1, s_2, ..., s_t) = f'(\overline{f}(s_1, ..., s_{t''}), \overline{f}(s_{t''+1}, ..., s_t))$ for every $t'' \in [t-1]$.

We prove Eq. (4.2) by induction on t. Assuming, without loss of generality that $t'' < t'$, we first establish Eq. (4.2) for the case of $t'' = t' - 1 \in [t-1]$. In this case, we have

$$\begin{aligned} f'(\overline{f}(s_1, ..., s_{t'}), \overline{f}(s_{t'+1} ..., s_t))) &= f'(f'(\overline{f}(s_1, ..., s_{t'-1}), s_{t'}), \overline{f}(s_{t'+1} ..., s_t))) \\ &= f'(\overline{f}(s_1, ..., s_{t'-1}), f'(s_{t'}, \overline{f}(s_{t'+1} ..., s_t))) \\ &= f'(\overline{f}(s_1, ..., s_{t'-1}), \overline{f}(s_{t'}, s_{t'+1} ..., s_t)), \end{aligned}$$

where the second equality uses the claim's hypothesis (with $x = \overline{f}(s_1, ..., s_{t'-1})$, $y = s_{t'}$ and $z = \overline{f}(s_{t'+1} ..., s_t)$) and the other equalities use the induction hypothesis. The general case (of $t'' < t'$) is handled by $t' - t''$ iterations of the foregoing special case (i.e., going from t' to $t' - 1$, then to $t' - 2$, and similarly all the way till $t' - (t' - t'')$).

Having established Eq. (4.2), we turn to the actual claim. Recalling that any $w \in [n]$ can be written as $\overline{f}(E(w, 1), ..., E(w, \ell)) = \overline{f}(s_1^w, ..., s_\ell^w)$, we establish $f'(x, f'(y, z)) = f'(f'(x, y), z)$ by using

$$\begin{aligned} f'(\overline{f}(s_1^x, ..., s_\ell^x), f'(\overline{f}(s_1^y ..., s_\ell^y), \overline{f}(s_1^z ..., s_\ell^z))) &= f'(\overline{f}(s_1^x, ..., s_\ell^x), \overline{f}(s_1^y ..., s_\ell^y, s_1^z ..., s_\ell^z)) \\ &= f'(\overline{f}(s_1^x, ..., s_\ell^x, s_1^y ..., s_\ell^y), \overline{f}(s_1^z ..., s_\ell^z)) \\ &= f'(f'(\overline{f}(s_1^x, ..., s_\ell^x), \overline{f}(s_1^y ..., s_\ell^y)), \overline{f}(s_1^z ..., s_\ell^z)). \end{aligned}$$

This completes the proof of the claim. ■

Conclusion: As stated upfront, if f represents a group, then, with high probability, Constuction 4.3.1 returns $f' = f$, and in this case our tester accepts. On the other hand, if f is ϵ-far from representing a group, then either Constuction 4.3.1 rejects or it returns f' that is either ϵ-far from f or does not represent a group. In each of these cases, our tester rejects (with high probability in case f' is ϵ-far from f). ■

Digest. The proof of Theorem 4.3.2 reduces testing a function $f : [n] \times [n] \to [n]$, which supposedly represents a group, to checking a data structure of size $\widetilde{O}(n)$. Recall that the latter data structure is created by Constuction 4.3.1, which uses $\widetilde{O}(n)$ queries to f (and $\widetilde{O}(n)$ times). When f represents a group, the data structure allows to compute f in logarithmic time (without access to f itself). In general, this data structure yields a function f', and we can test whether $f' = f$ and reduce testing properties of f to checking properties of f'. In particular, we tested whether f is a group by checking f' represents a group (and $f' = f$). Likewise, we can test whether f represents an Abelian group by checking whether f' represents an Abelian group.

4.4 Main result: Testing whether f represents a group, take 3

In this section, we present a tester of sub-linear (in n^2) time complexity; specifically, the tester will have time complexity $\widetilde{O}(n/\epsilon)$. Here it is instructive to view $f : [n] \times [n] \to [n]$ as an n-by-n matrix (or table). Intuitively, Steps 1 and 2 in Construction 4.4.1 constitute a test of cancellativity, whereas Step 3 is a test of associativity. Recall that f represents a group if and only if it is both cancellative (i.e., $f(x, y) = f(x, y')$ implies $y = y'$ and $f(x, y) = f(x', y)$ implies $x = x'$) and associative (i.e., $f(x, f(y, z)) = f(f(x, y), z)$).

We comment that Steps 1 and 2 in Construction 4.4.1 replace a less efficient test that appears in [7, Sec. 4.3], which checks for *collisions* in *each* row of f. In contrast, Step 3 in Construction 4.4.1 is identical to the tester that appears in [7, Sec. 4.1.1]. (Our analysis utilizes all three steps, but it may be that Step 2 is actually unnecessary.)

Construction 4.4.1 (a direct tester for the group property): *On input $n \in \mathbb{N}$ and $\epsilon > 0$, and oracle access to $f : [n] \times [n] \to [n]$, viewed as an n-by-n matrix over $[n]$, the tester proceeds as follows.*

1. Check that random rows and columns contains permutations: *Select uniform at random $O(1/\epsilon)$ elements $r \in [n]$, and verify that for each of these r's the mappings $x \mapsto f(r, x)$ and $x \mapsto f(x, r)$ are permutations. Otherwise, reject.*

2. Check that random rectangles consist of columns that contain no repeated values:[5] *For each $i \in [\log_2 n]$, select uniformly at random a set of $O(n/2^i)$ rows and a set of $O((2^i \log n)/\epsilon)$ columns and check that each value appears at most once in each of the residual columns. Otherwise, reject.* (That is, if for one of the selected columns c and two selected rows $r_1 \neq r_2$ it holds that $f(r_1, c) = f(r_2, c)$, then reject.)

3. Checking associativity. *Specifically, for random values of two of the three variables, we check the associativity condition for all values of the remaining variable. This is done for each of the possible three choices of the identity of the remaining variable.*

   Repeat each of the following three checks $O(1/\epsilon)$ times.

   (a) Checking all values of the first variable: *Select uniformly at random $r, s \in [n]$, and verify that $f(a, f(r, s)) = f(f(a, r), s)$ holds for every $a \in [n]$.*

   (b) Checking all values of the second variable: *Select uniformly at random $r, s \in [n]$, and verify that $f(r, f(a, s)) = f(f(r, a), s)$ holds for every $a \in [n]$.*

   (c) Checking all values of the third variable: *Select uniformly at random $r, s \in [n]$, and verify that $f(r, f(s, a)) = f(f(r, s), a)$ holds for every $a \in [n]$.*

   Reject if any of these checks fails.

If none of the steps rejected, then accept.

The time complexity of the foregoing tester is $O(n/\epsilon) + O(\epsilon^{-1} \cdot n \log^2 n) + O(n/\epsilon) = O(\epsilon^{-1} \cdot n \log^2 n)$, and it always accepts if f represents a group operation. We shall show that, if the tester accepts with probability at least 1/3, then f is ϵ-close to representing a group operation.

[5]Indeed, it is natural to perform the same check for rows, but this is unnecessary for our analysis. Performing both checks covers Step 1 as a special case for $i = \log_2 n$ and $i = 1$, respectively.

Theorem 4.4.2 (Construction 4.4.1 rejects (whp) functions that are ϵ-far from respresenting a group): *If Construction 4.4.1 accepts f with probability at least $1/3$, then f is ϵ-close to representing a group operation.*

Proof: Using a sufficiently small $\epsilon' = \Omega(\epsilon)$, we show that if Construction 4.4.1 accepts f with probability at least 1/3, then f is $O(\epsilon')$-close to a self-corrected version (defined in Eq. (4.3) below), which in turn represents a group operation. Specifically, after establishing a few preliminary claims, we shall show that the latter function is cancellative and associative, which implies that it describes a group.

Let R (resp., C) denote the set of r's such that $x \mapsto f(r,x)$ (resp., $x \mapsto f(x,r)$) is a permutation, and note that $|R| \geq (1-\epsilon') \cdot n$ (resp., $|C| \geq (1-\epsilon') \cdot n$), since otherwise Step 1 rejects with high probability. The lower bounds on $|R|$ and $|C|$ implies that each value occurs at most $\min(|R|,|C|)$ times in the rectangle $R \times C$, but it does not imply that each value occurs at least $\min(|R|,|C|)$ times. Nevertheless, a slightly weaker statement does hold.

Claim 4.4.2.1 ($R \times C$ contains at least $|R| - 2\epsilon' n$ occurrences of each value): *For every $v \in [n]$, it holds that $|\{(r,c) \in R \times C : f(r,c) = v\}| \geq |R| - 2\epsilon' n$.*

Note that if f is guaranteed to be cancellative, as in [7, Sec. 4.1], then $R = C = [n]$, and Claims 4.4.2.1 and 4.4.2.2 hold vacuously. In this case, Steps 1 and 2 of Construction 4.4.1 are unnecessary, and the time complexity of the tester is reduced to $O(n/\epsilon)$.

Proof: Fixing v and letting $\overline{C} \overset{\text{def}}{=} [n] \setminus C$, we shall prove that $|\{(r,c) \in [n] \times \overline{C} : f(r,c) = v\}| \leq 2\epsilon' n$, and the claim will follow (since each $r \in R$ contains the value v). For each $c \in \overline{C}$, we let n_c denote the number of v's in column c. Assuming, towards the contradiction that $\sum_{c \in \overline{C}} n_c > 2\epsilon' n$, we shall show that in such a case Step 2 rejects with high probability.

Letting $\ell = \log_2 n$, for each $i \in [\ell]$, we define $B_i = \{c \in \overline{C} : 2^i \leq n_c < 2^{i+1}\}$, and observe that $B_0 \overset{\text{def}}{=} \overline{C} \setminus \bigcup_{i \in [\ell]} B_i = \{c \in \overline{C} : n_c \leq 1\}$, whereas $\sum_{c \in B_0} n_c \leq |\overline{C}|$. Hence,

$$\sum_{i \in [\ell]} \sum_{c \in B_i} n_c \; > \; 2\epsilon' n - |\overline{C}| \; \geq \; \epsilon' n.$$

It follows that there exists $i \in [\ell]$ such that $\sum_{c \in B_i} n_c > \epsilon' n / \ell$. Fixing this i, observe that $|B_i| > \frac{\epsilon' n/\ell}{2^{i+1}} = \frac{\epsilon'}{2^{i+1}\ell} \cdot n$. Hence, with high probability, a random set of $O(2^i \ell/\epsilon)$ columns contains a column in B_i, denoted c. Recalling that $n_c \geq 2^i \geq 2$, with high probability, a random set of $O(n/2^i)$ rows contains two distinct rows r_1, r_2 such that $f(r_1,c) = v = f(r_2,c)$. But in such a case, Step 2 rejects. The claim follows. ■

A self-corrector for f. For any $v \in [n]$ and $r \in R$, let $c_v(r)$ be the unique value c that satisfies $f(r,c) = v$, where unique existence is guaranteed by $r \in R$ (which implies that v occurs exactly once in row r). Letting $a \odot b = f(a,b)$, we define a candidate self-correcting function $g : [n] \times [n] \to [n]$ as follows[6]

$$g(x,y) \overset{\text{def}}{=} \text{Plurality}_{r \in R}\{(x \odot r) \odot c_y(r)\}. \tag{4.3}$$

Indeed, for every $x, y \in [n]$ and $r \in R$, it holds that $x \odot (r \odot c_y(r)) = x \odot y$, but f is not necessarily associative (i.e., $(x \odot r) \odot c_y(r)$ may not equal $x \odot (r \odot c_y(r))$). Nevertheless, since F is "close to being associative" (per being rarely rejected by Step 3), one may hope that $(x \odot r) \odot c_y(r) = x \odot (r \odot c_y(r))$ typically holds (i.e., holds for $1 - O(\epsilon')$ fraction of the r's in R). This is indeed proved in Claim 4.4.2.3, but our proof (as well as other proofs that follow) uses the following claim.

Claim 4.4.2.2 (c_v is almost uniformly distributed in $[n]$): *For every $v \in [n]$, when selecting uniformly $r \in R$, it holds that $c_v(r)$ is $4\epsilon'$-close to the uniform distribution over $[n]$.*

[6]It seems that a more natural (and "symmetric") form of a self-corrector is given by

$$\text{Plurality}_{r,s \in R}\{s \odot ((c_x(s) \odot r) \odot c_y(r))\}.$$

However, as shown in the auxiliary claim of Claim 4.4.2.5, this more complicated form is equivalent to the one we use below (i.e., in Eq. (4.3)).

Proof: Fixing v, we call $r \in R$ good if $c_v(r) \in C$, and observe that good r's yield different $c_v(r)$'s (because $c \stackrel{\rm def}{=} c_v(r) = c_v(r')$ imply $r \odot c = v = r' \odot c$, which implies $r = r'$ since $c \in C$). Recalling that (by Claim 4.4.2.1) the number of v-entries in columns that is not in C is at most $2\epsilon' \cdot n$, it follows that at most $2\epsilon' n$ rows in R have a v-entry in a column that is not in C, Hence,

$$\Pr_{r\in R}[c_v(r) \in C] \geq \frac{|R| - 2\epsilon' n}{|R|} > 1 - 3\epsilon';$$

that is, $r \in R$ is good with probability at least $1 - 3\epsilon'$. Recalling that different good r's (in R) have different values of $c_v(r)$, it follows that, for a uniformly distributed $r \in R$, the distribution of $c_v(r)$ is $3\epsilon'$-close to be uniform over some set of $|R|$ values. The claim follows. ■

Claim 4.4.2.3 (g is supported by a strong majority): *For every $a, b \in [n]$,*

$$\Pr_{r,r'\in R}\left[(a \odot r) \odot c_b(r) = (a \odot r') \odot c_b(r')\right] = 1 - O(\epsilon'). \tag{4.4}$$

Equivalently, for every $a, b \in [n]$, it holds that $\Pr_{r\in R}[(a \odot r) \odot c_b(r) = g(a, b)] = 1 - O(\epsilon')$.

The following proof (as well as subsequent ones) makes essential use of the specific way in which associativity is checked in Step 3. Specifically, the fact that (w.h.p.) none of the invocations of Step 3a rejects implies that

$$\Pr_{r,s\in[n]}\left[(\forall a \in [n])\, f(a, f(r, s)) = f(f(a, r), s)\right] \; > \; 1 - \epsilon'. \tag{4.5}$$

We shall use the fact that Eq. (4.5) implies that, for every function $\alpha : [n] \times [n] \to [n]$, it holds that

$$\Pr_{r,s\in[n]}\left[f(\alpha(r, s), f(r, s)) = f(f(\alpha(r, s), r), s)\right] \; > \; 1 - \epsilon'. \tag{4.6}$$

Analogously, by Steps 3b and 3c, we get

$$\Pr_{r,s\in[n]}\left[f(r, f(\alpha(r, s), s)) = f(f(r, \alpha(r, s)), s)\right] \;\; > \;\; 1 - \epsilon' \tag{4.7}$$

$$\Pr_{r,s\in[n]}\left[f(r, f(s, \alpha(r, s)))) = f(f(r, s), \alpha(r, s))\right] \;\; > \;\; 1 - \epsilon'. \tag{4.8}$$

Proof: To gain some intuition, note that Eq. (4.8) immediately implies that Eq. (4.4) holds for any fixed $b \in [n]$ and a uniformly distributed $a \in [n]$. Furthermore, fixing $a \in [n]$ and considering a random $b \in [n]$, one may use Claim 4.4.2.1 to show that $c_b(r)$ is distributed almost independently of r, and then derive Eq. (4.4) by using Eq. (4.6). The actual claim, which refers to any $a, b \in [n]$, is proved by a more complicated argument, which employs the foregoing ideas more extensively.

Fixing any $a, b \in [n]$, we consider r and r' that are distributed independently and uniformly in R. Then, by Claim 4.4.2.2, we have $\Pr_{r\in R}[c_b(r) \in C] \geq 1 - 4\epsilon'$, whereas $c_b(r')$ is defined per $r' \in R$. In this case (i.e., $c_b(r) \in C$), there exists a unique $d = d_b(r, r')$ such that $d \odot c_b(r) = c_b(r')$. Hence, it holds that

$$\Pr_{r,r'\in R}[r' \odot c_b(r') = r' \odot (d_b(r, r') \odot c_b(r))] \; \geq \; 1 - 4\epsilon'. \tag{4.9}$$

Recalling that $c_b(r)$ is $4\epsilon'$-close to uniform (and r' is ϵ'-close to uniform), by Eq. (4.7) we have

$$\Pr_{r,r'\in R}\left[r' \odot (d_b(r, r') \odot c_b(r)) = (r' \odot d_b(r, r')) \odot c_b(r)\right] \; \geq \; 1 - 6\epsilon'. \tag{4.10}$$

Combining $r \odot c_b(r) = r' \odot c_b(r')$ with Eq. (4.9)&(4.10), we get

$$\Pr_{r,r'\in R}\left[r \odot c_b(r) = (r' \odot d_b(r, r')) \odot c_b(r)\right] \; \geq \; 1 - 10 \cdot \epsilon'. \tag{4.11}$$

Recalling that $c_b(r) \in C$ means that there is a unique value w such that $w \odot c_b(r) = b$ (i.e., $w = r$), it follows that $\Pr_{r,r'\in R}[r = r' \odot d_b(r, r')] \; = \; 1 - O(\epsilon')$. Hence, for every $a, b \in [n]$, we have

$$\Pr_{r,r'\in R}\left[(a \odot r) \odot c_b(r) = (a \odot (r' \odot d_b(r, r'))) \odot c_b(r)\right] \; = \; 1 - O(\epsilon'). \tag{4.12}$$

Assuming that $(a\odot(r'\odot d_b(r,r')))\odot c_b(r)$ equals $(a\odot r')\odot(d_b(r,r')\odot c_b(r))$, which in turn equals $(a\odot r')\odot c_b(r')$, would have established the claim; that is, under this assumption, Eq. (4.12) implies $\Pr_{r,r'\in R}[(a\odot r)\odot c_b(r) = (a\odot r')\odot c_b(r')] = 1-O(\epsilon')$. However, associativity is only guaranteed by Eq. (4.6)–(4.8) *when two of the three operands are uniformly and independently distributed.* Thus, towards applying the associativity guaranteee, we first argue that the joint distributions of $(r', d_b(r,r'))$ and $(d_b(r,r'), c_b(r))$ are (each) almost uniformly distributed in $[n]\times[n]$.

Recall that, for $r',r\in R$ such that $c_b(r)\in C$, it holds that $d_b(r,r')\odot c_b(r) = c_b(r')$ uniquely determines $d_b(r,r')$ as a function of $c_b(r)$ and $c_b(r')$. Considering uniformly distributed $r,r'\in R$, we show that the distribution of $d_b(r,r')$ is almost uniform (over $[n]$) and independent of the distribution of r' (resp., $c_b(r)$).

- Fixing an arbitrary $r'\in R$ and letting r be uniformly distributed in R, we wish to show that $d_b(r,r')$ is almost uniformly distributed in $[n]$.

  We first recall that (by Claim 4.4.2.1) the rectangle $R\times C$ contains at least $n' \stackrel{\rm def}{=} |R|-2\epsilon' n \geq (1-3\epsilon')\cdot n$ entries that hold the value $c_b(r')$, whereas each row in R (resp., column in C) contains a single $c_b(r')$ value. Hence, $R\times C$ contains an n'-by-n' square, denoted $R'\times C'$, such that each row in R' (resp., column in C') contains a single $c_b(r')$ value that appears in a column in C' (resp., a row in R'). It follows that, for a uniformly distributed $c\in C'$, the unique d that satisfies $d\odot c = c_b(r')$ is uniformly distributed in R'.

  Next, we note that distinct r's in R such that $c_b(r)\in C$ must have distinct $c_b(r)$'s values. Hence, for a random $r\in R$ such that $c_b(r)\in C$, the value $c_b(r)$ is uniformly distributed in $S \stackrel{\rm def}{=} \{c_b(u)\in C : u\in R\}$. Using the conclusion of the previous paragraph, it follows that, for a random $r\in R$ such that $c_b(r)\in S\cap C'$, the value $d_b(r,r')$ (which solves $d\odot c_b(r) = c_b(r')$) is uniformly distributed in some $|S\cap C'|$-subset of $[n]$. Using $\Pr_{r\in R}[c_b(r)\in S\cap C'] = 1-O(\epsilon')$, it follows that $d_b(r,r')$ is almost uniformly distributed in $[n]$.

- Fixing an arbitrary $r\in R$ such that $c_b(r)\in C$ and letting r' be uniformly distributed in R, we wish to show that $d_b(r,r')$ is almost uniformly distributed in $[n]$.

  In this case (i.e., $c_b(r)\in C$), the value of $d_b(r,r')$ is determined by $c_b(r')$, which is almost uniformly distributed in $[n]$, such that different values of $c_b(r')$ yield differnt values of $d_b(r,r')$ (since the column $c_b(r)\in C$ contains different values).

Having estabished that, for uniformly distributed $r,r'\in R$, the distribution of $d_b(r,r')$ is almost uniform (over $[n]$) and independent of the distribution of r' (resp., $c_b(r)$), we observe that, for every $a,b\in[n]$, with probability $1-O(\epsilon')$ over the choice of $r,r'\in$R, it holds that

$$\begin{aligned}(a\odot(r'\odot d_b(r,r'))\odot c_b(r) &= ((a\odot r')\odot d_b(r,r'))\odot c_b(r)\\ &= (a\odot r')\odot(d_b(r,r')\odot c_b(r))\\ &= (a\odot r')\odot c_b(r')\end{aligned}$$

where the first (resp., second) equality uses Eq. (4.6) and relies on the fact that r' and $d_b(r,r')$ (resp., $d_b(r,r')$ and $c_b(r)$) are almost independently distributed in $[n]$. Hence, we have

$$\Pr_{r,r'\in R}\left[(a\odot(r'\odot d_b(r,r')))\odot c_b(r) = (a\odot r')\odot c_b(r')\right] \;\geq\; 1-O(\epsilon'). \tag{4.13}$$

Combining Eq. (4.12) with Eq. (4.13), the claim follows. ■

Towards the key claims. Using the foregoing claims, we prove the key features of g; that is, that it is close to f, cancellative, and associative. (Recall that $f(a,b)=a\odot b$.)

Claim 4.4.2.4 (g is $O(\epsilon')$-close to f): $\Pr_{a,b\in[n]}[g(a,b) = f(a,b)] = 1-O(\epsilon')$. *Furthermore, for every* $b\in[n]$*, it holds that* $\Pr_{a\in[n]}[g(a,b)=f(a,b)] = 1-O(\epsilon')$.

Proof: We shall prove the furthermore claim; that is, for every $b \in [n]$, we shall show that $g(\cdot,b)$ is $O(\epsilon')$-close to $f(\cdot,b)$. This is proved by combining the following two facts.

- Using Claim 4.4.2.3, *for every* $a,b \in [n]$, *we have* $\Pr_{r\in R}[g(a,b) = f(f(a,r),c_b(r))] = 1-O(\epsilon')$, *where* $f(r,c_b(r)) = b$.
- Using Eq. (4.8), *for every* $b \in [n]$, *we have* $\Pr_{a\in[n],r\in R}[f(f(a,r),c_b(r)) = f(a,f(r,c_b(r)))] > 1-\epsilon'$.

Combining these two facts (and recalling that $f(r,c_b(r)) = b$), for every $b \in [n]$, we have $\Pr_{a\in[n]}[g(a,b) = f(a,b)] = 1-O(\epsilon')$. ■

Claim 4.4.2.5 (g is cancellative):

1. *For every* $a,a',b \in [n]$, *if* $g(a,b) = g(a',b)$, *then* $a = a'$.
2. *For every* $a,b,b' \in [n]$, *if* $g(a,b) = g(a,b')$, *then* $b = b'$.

Proof: We start with Part 1. By Claim 4.4.2.3, $\Pr_{r\in R}[(a\odot r)\odot c_b(r) = g(a,b)] = 1-O(\epsilon')$, and the same holds for a'. Hence, $g(a,b) = g(a',b)$ implies

$$\Pr_{r\in R}\left[(a\odot r)\odot c_b(r) = (a'\odot r)\odot c_b(r)\right] = 1-O(\epsilon'). \tag{4.14}$$

Recalling that $\Pr_{r\in R}[c_b(r)\in C] = 1-O(\epsilon')$ and that in this case there is a unique w such that $w\odot c_b(r) = b$, it follows that

$$\Pr_{r\in R}\left[(a\odot r) = (a'\odot r)\right] = 1-O(\epsilon') > \epsilon',$$

which implies that $a = a'$ (since $a\odot r = a'\odot r$ for some $r \in C$).

Turning to Part 2, we first establish it for $a \in R$. Analogously to Eq. (4.14), we observe that $g(a,b) = g(a,b')$ implies

$$\Pr_{r\in R}\left[(a\odot r)\odot c_b(r) = (a\odot r)\odot c_{b'}(r)\right] = 1-O(\epsilon'). \tag{4.15}$$

Now, if $a \in R$, then $\Pr_{r\in R}[a\odot r \in R] \geq \frac{|R|-(n-|R|)}{|R|} > 1-2\epsilon'$ follows (since each element appears once in row a). In this case, $c_b(r) = c_{b'}(r)$ follows for some $r \in R$ (which satisfies $a\odot r \in R$), which implies $r\odot c_b(r) = r\odot c_{b'}(r)$, and it follows that $b = b'$ (since $r\odot c_b(r) = b$ and ditto for b'). We now turn to the general case (of $a \in [n]$), but first prove an auxiliary claim.

Auxliliary claim: For every $x,y \in [n]$, it holds that $\Pr_{r\in R}[r\odot g(c_x(r),y) = g(x,y)] = 1-O(\epsilon')$.

Proof: Fixing $x,y \in [n]$, we observe that, with probability $1-O(\epsilon')$ over the choices of $r,s \in R$, it holds that

$$\begin{aligned} r\odot g(c_x(r),y) &= r\odot((c_x(r)\odot s)\odot c_y(s)) \\ &= (r\odot(c_x(r)\odot s))\odot c_y(s) \\ &= ((r\odot c_x(r))\odot s)\odot c_y(s) \\ &= (x\odot s)\odot c_y(s) \\ &= g(x,y) \end{aligned}$$

where the first and last equality are due to Claim 4.4.2.3, whereas the second and third equalities are guranateed by Eq. (4.7) (and Claim 4.4.2.2). This completes the proof of the auxliliary claim.

The general case of Part 2. For any $a,b,b' \in [n]$, if $g(a,b) = g(a,b')$, then (by the auxiliary claim) we have

$$\Pr_{r\in R}\left[r\odot g(c_a(r),b) = r\odot g(c_a(r),b')\right] = 1-O(\epsilon').$$

Using the hypothesis that $r \in R$, this implies

$$\Pr_{r\in R}\left[g(c_a(r),b) = g(c_a(r),b')\right] = 1-O(\epsilon').$$

Observing that (by Claim 4.4.2.2) it holds that $\Pr_{r\in R}[c_a(r) \in R] = 1-O(\epsilon')$ and applying the foregoing special case (which refers to a first operand in R), we infer that $b = b'$. ■

Claim 4.4.2.6 (g is associative): *For every $a,b,c \in [n]$, it holds that $g(a,g(b,c)) = g(g(a,b),c)$.*

Proof: We prove the claim in two steps, first establishing it only for a random $a \in [n]$, and then inferring that it holds for any $a \in [n]$. Specifically, we first prove that, for every $y,z \in [n]$,

$$\Pr_{r\in R}\left[g(r,g(y,z)) = g(g(r,y),z)\right] = 1 - O(\epsilon'). \tag{4.16}$$

Fixing $y,z \in [n]$ and using the furthermore part of Claim 4.4.2.4 (as well as Claim 4.4.2.3), we observe that with probability $1-O(\epsilon')$ over the choice of $r,s \in R$, it holds that

$$\begin{aligned} g(r,g(y,z)) &= f(r,g(y,z)) \\ &= f(r,(y\odot s)\odot c_z(s)). \end{aligned}$$

Hence,

$$\Pr_{r,s\in R}\left[g(r,g(y,z)) = r\odot((y\odot s)\odot c_z(s)))\right] = 1 - O(\epsilon'). \tag{4.17}$$

Next, using Eq. (4.7) (as well as Claim 4.4.2.2), with probability $1-O(\epsilon')$ over the choice of $r,s \in R$, it holds that

$$\begin{aligned} r\odot((y\odot s)\odot c_z(s)) &= (r\odot(y\odot s))\odot c_z(s) \\ &= ((r\odot y)\odot s)\odot c_z(s) \\ &= g(f(r,y),z) \\ &= g(g(r,y),z) \end{aligned}$$

where the first and second equalities are guranateed by Eq. (4.7) (while relying on the randomness of r and $c_z(s)$ (resp., r and s)), whereas the third equality use Claim 4.4.2.3, and the last equality uses the furthermore part of Claim 4.4.2.4. Hence,

$$\Pr_{r,s\in R}\left[r\odot((y\odot s)\odot c_z(s))) = g(g(r,y),z)\right] = 1 - O(\epsilon'). \tag{4.18}$$

Combining Eq. (4.17)&(4.18), we establish Eq. (4.16).

Using Eq. (4.16), we now establish the actual claim. In particular, we shall apply Eq. (4.16) four times, while observing that in each of these application the first operand is uniformly (or almost uniformly) distributed in R. Specifically, we observe that, for every $x,y,z \in [n]$, with probability $1-O(\epsilon')$ over the choice of $r \in R$, it holds that

$$\begin{aligned} g(r,g(x,g(y,z))) &= g(g(r,x),g(y,z)) \\ &= g(g(g(r,x),y),z) \\ &= g(g(r,g(x,y)),z) \\ &= g(r,g(g(x,y),z)), \end{aligned}$$

where the second transition we rely on the randomness of $g(r,x)$, which follows from Part 1 of Claim 4.4.2.5 (i.e., for every x, the mapping $r \mapsto g(r,x)$ is a bijection). Hence,

$$\Pr_{r\in R}\left[g(r,g(x,g(y,z))) = g(r,g(g(x,y),z))\right] = 1 - O(\epsilon'). \tag{4.19}$$

Using Part 2 of Claim 4.4.2.5, it follows that $g(x,g(y,z)) = g(g(x,y),z)$. ■

Conclusion. Claim 4.4.2.4 asserts that f is ϵ-close to g, whereas Claims 4.4.2.5 and 4.4.2.6 assert that g is cancellative and associative. Hence, g describes a finite cancellative semigroup, which means that g describes a group, and it follows that f is ϵ-close to representing a group. ■

Corollary 4.4.3 (testing the property of being a group) (Theorem 4.1.2, restated): *There exists a $(\widetilde{O}(n)/\epsilon)$-time tester for the property of being a group. Furthermore, the tester has one-sided error.*

Theorem 4.4.4 (testing the property of being an Abelian group): *There exists a $(\widetilde{O}(n)/\epsilon)$-time tester for the property of being an Abelian group. Furthermore, the tester has one-sided error.*

Proof: Our tester first invokes Construction 4.4.1 and rejects if it has rejected. Otherwise, it checks that $g : [n] \times [n] \to [n]$ as defined in Eq. (4.3) represents an Abelian group by selecting a random pair of elements in $[n]$ and checking whether they commute (under g). Here we use the well known fact that asserts that if a group is not Abelian, then at least 3/8 of the element pairs do not commute.[7]

Note that g can be evaluated, at random, by making $2+n$ queries to f (i.e., $g(x,y) \leftarrow f(f(x,r), c_y(r))$ for a uniformly chosen $r \in [n]$, where $c_y(r)$ is computed by finding c such that $f(r,c) = y$), and recall that if f represents a group then Construction 4.4.1 always accepts and $g = f$. Hence, our tester always accepts when f represents an Abelian group. Turning to the case that f is ϵ-far from representing an Abelian group, we prove that our tester rejects with probability at least 1/3 (and obtain the desired tester by error reduction).

We actually prove the counterpositive. Assuming that f is accepted with probability at least 2/3, it follows (by Theorem 4.4.2) that f is ϵ-close to representing a group. Furthermore, in that case (by Claim 4.4.2.3), the random evaluation of g yields its correct value, with probability at least $1 - O(\epsilon)$, whereas g is ϵ-close to f (see Claim 4.4.2.4) and represents a group (see Claims 4.4.2.5 and 4.4.2.6). Lastly, note that the group represented by g must be Abelian, because otherwise our tester would have rejected with probability at least $(1 - O(\epsilon)) \cdot 3/8 > 1/3$. ■

4.5 A lower bound

Theorem 4.5.1 (a lower bound on testing properties of groups (Theorem 4.1.6, restated)): *Let $\mathcal{G}$ be a set of finite groups that is closed under isomorphism and contains all cyclic groups of prime order. Then, testing $\mathcal{G}$ requires $\Omega(\log n)$ queries.*

Proof Sketch: We employ the "indistinguishability technique" (see [11, Sec. 7.2]). Using two primes $m, n \in \mathbb{N}$ such that $m < n < m + o(m/\log m)$, we consider the following two distributions on functions from $[n] \times [n]$ to $[n]$.

1. The distribution of functions that represent random isomorphic copies of the cyclic group $\mathbb{Z}_n$; that is, the distribution is uniform over the functions $f : [n] \times [n] \to [n]$ that represent a cyclic group over $[n]$.

   To streamline the analysis, we also select uniformly at random an m-subset $S \subset [n]$.

2. The distribution of functions that consist of functions that represent a cyclic group with m elements augmented by an arbitrary function (e.g., the all-1 function) on the rest of the domain; that is, the distribution is uniform over the functions $f : [n] \times [n] \to [n]$ such that for some m-subset S it holds that $f|_S$ represent a cyclic group over S whereas $f(x,y) = 1$ for every $(x,y) \in [n]^2 \setminus S^2$.

We first show that one cannot distinguish these two distributions when making $\ell = o(\log n)$ queries. This is the case because distinguishing is possible only either when making a query in $[n]^2 \setminus S^2$, which happens with probability at most $2\ell \cdot (n-m)/n$, or when encountering the neutral element of the relevant group.[8] Observing that each query (in S^n) consists of a pair in which each element is either a random element of the relevant group or the answer of a prior query, it follows that each answer to each query is a linear combination

[7] An elementrary proof of a weaker lower bound (of 1/4) proceeds as follows. Suppose that more than 3/4 of the pairs are commuting. Then, for every $a \in [n]$,

$$\Pr_{r,s\in[n]}[r \odot (a \odot s) \neq (a \odot s) \odot r] < 1/4 \quad \text{and} \quad \Pr_{r,s\in[n]}[a \odot (s \odot r)) \neq a \odot (r \odot s)] < 1/4$$

which implies $\Pr_{r,s\in[n]}[(r \odot a) \odot s \neq (a \odot r) \odot s] < 1/2$. Hence, $\Pr_{r\in[n]}[r \odot a \neq a \odot r] < 1/2$ holds for every $a \in [n]$. It follows that, for every $a, b \in [n]$,

$$\Pr_{r\in[n]}[a \odot (b \odot r) \neq (b \odot r) \odot a] < 1/2 \quad \text{and} \quad \Pr_{r\in[n]}[b \odot (r \odot a) \neq b \odot (a \odot r)] < 1/2$$

which implies $\Pr_{r\in[n]}[(a \odot b) \odot r \neq (b \odot a) \odot r)] < 1$. Hence, $a \odot b = b \odot a$ holds for every $a, b \in [n]$.

[8] The point is that all (new) answers look random, until one queries a pair that contains the neutral element (in which case one get back the other element).

of random elements of the group such that each coefficient is at most 2^ℓ. Hence, the latter answer is unlikely to be the neutral element, because this would require a coefficient that is a multiple of the group size (since the group is of prime order).[9]

On the other hand, each function in the first distribution represents a cyclic group of order n. In contrast, we shall show that each function in the second distribution is 0.222-far from representing a cyclic group of order n. The latter fact is shown by using the following claim, which is a special case of [19, Cor. 1].

Claim (see [21, Lem. 3]): Let $(S, \odot)$ and $(S', \odot')$ be two groups such that $|S| \leq |S'|$. If $(S, \odot)$ is not isomorphic to any subgroup of $(S', \odot')$, then for every injective map $\mu : S \to S'$ it holds that

$$\mathrm{Pr}_{x,y \in S}[\mu(x \odot y) = \mu(x) \odot' \mu(y)] \ \leq \ \frac{7}{9}$$

In particular, we let $(S', \odot')$ be an arbitrary group with n elements, which is necessarily cyclic (since n is a prime), and let f be selected from the second distribution, where S is the corresponding p-subset and $\odot$ represents the restriction of f to S^2 (i.e., $(S, \odot)$ be a cyclic group of order m). Then, for every injective map $\mu : S \to S'$, the function $\mu \circ f$ and the function $f' \circ \mu$ such that f' represents $\odot'$ differ on at least $\frac{2}{9} \cdot m^2$ elements. Using $n = (1 + o(1)) \cdot m$, it follows that f is 0.222-far from f'. ■

Corollary 4.5.2 (lower bounds on specific testing tasks):

- *Testing the property of being a group requires* $\Omega(\log n)$ *queries.*
- *Testing the property of being an Abelian group requires* $\Omega(\log n)$ *queries.*
- *Testing the property of being a cyclic group requires* $\Omega(\log n)$ *queries.*

[9]Each query has the form (x, y), where x and y are prior answers (or new elements selected in $[n]$), whereas the answer is $f(x, y)$, which is typically the "sum" of the two elements (in the relevant group). Hence, the answer to the i^{th} query is a linear combination of random elements with coefficients that are at most 2^i.

ACKNOWLEDGMENTS

I wish to express my deep gratitude to my advisor, Oded, for his caring mentorship, support throughout my masters, as well as providing me with full academic freedom. I am also grateful to Alex Lubotzky for inspiring discussions.

BIBLIOGRAPHY

[1] N. Alon, I. Benjamini, E. Lubetzky, and S. Sodin. Non-Backtracking Random Walks Mix Faster. *Communications in Contemporary Mathematics*, Vol. 9 (4), pages 585–603, 2007.

[2] N. Alon and Y. Roichman. Random Cayley graphs and expanders. *Random Structures Algorithms*, Vol. 5 (2), pages 271–284, 1994.

[3] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, Vol. 36 (4), pages 889–974, 2006. Extended abstract in *36th STOC*, 2004.

[4] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Science*, Vol. 47, No. 3, pages 549–595, 1993.

[5] B. Bollobas. A Probabilistic Proof of an Asymptotic Formula for the Number of Labelled Regular Graphs. *European Journal of Combinatorics*, Vol. 1, 311–316, 1980.

[6] B. Bollobas. The Isoperimetric Number of Random Regular Graphs. *European Journal of Combinatorics*, Vol. 9, 241–244, 1988.

[7] F. Ergun, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-Checkers. *Journal of Computer and System Science*, Vol. 60 (3), pages 717–751, 2000.

[8] S. Evra, S. Gadot, O. Klein, abd I. Komargodski. Verifying Groups in Linear Time. *ECCC*, TR23-195, 2023.

[9] E. Fischer and A. Matsliah. Testing Graph Isomorphism. *SIAM Journal on Computing*, Vol. 38 (1), pages 207–225, 2008.

[10] K. Friedl, G. Ivanyos, and M. Santha. Efficient testing of groups. In *37th ACM Symposium on the Theory of Computing*, pages 157–166, 2005.

[11] O. Goldreich. *Introduction to Property Testing.* Cambridge University Press, 2017.

[12] O. Goldreich. Hierarchy Theorems for Testing Properties in Size-Oblivious Query Complexity. *Comput. Complex.*, Vol. 28 (4), pages 709–747, 2019.

[13] O. Goldreich. Testing Isomorphism in the Bounded-Degree Graph Model. *ECCC*, TR19-102, 2019.

[14] O. Goldreich. Robust Self-Ordering versus Local Self-Ordering. *ECCC*, TR21-034, 2021.

[15] O. Goldreich, M. Krivelevich, I. Newman, and E. Rozenberg. Hierarchy Theorems for Property Testing. *Comput. Complex.*, Vol. 21 (1), pages 129–192, 2012.

[16] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, pages 302–343, 2002. Extended abstract in *29th STOC*, 1997.

[17] O. Goldreich and L. Tauber. Testing in the bounded-degree graph model with degree bound two. *ECCC*, TR22-184, 2022.

[18] O. Goldreich and A. Wigderson. Robustly Self-Ordered Graphs: Constructions and Applications to Property Testing. *TheoretiCS*, Vol. 1, Art. 1, 2022.

[19] G, Ivanyos, F. Le Gall and Y. Yoshida. On the distance between non-isomorphic groups. `arXiv:1107.0133` [math.GR].

[20] M. Kusumoto and Y. Yoshida. Testing Forest-Isomorphism in the Adjacency List Model. In *Int. Colloquium on Automata, Languages and Programming*, pages 763–774, LNCS 8572, 2014.

[21] F. Le Gall and Y. Yoshida. Property testing for cyclic groups and beyond. *J. Comb. Opt.*, Vol. 26, pages 636–654, 2013.

[22] A. McIver and P.M. Neumann. Enumerating finite groups. *Quart. J. Math. Oxford*, Vol. 38, pages 473–488, 1987.

[23] E. Mossel and N. Sun. Shotgun assembly of random regular graphs. `arXiv:1512.08473` [math.PR], 2015.

[24] P.M. Neumann. An enumeration theorem for finite groups. *Quart. J. Math. Oxford*, Vol. 20, pages 395–401, 1969.

[25] I. Newman and C. Sohler. Every Property of Hyperfinite Graphs Is Testable. *SIAM Journal on Computing*, Vol. 42 (3), pages 1095–1112, 2013.

[26] L. Pyber. Enumerating finite groups of given order. *Ann. of Math.*, Vol. 137 (12), pages 203–220, 1993.

[27] S. Rajagopalan and L.J. Schulman. Verification of Identities. *SIAM Journal on Computing*, Vol. 29 (4), pages 1155–1163, 2000.

[28] G. Rothblum, S. Vadhan, and A. Wigderson. Interactive Proofs of Proximity: Delegating Computation in Sublinear Time. In *45th ACM Symposium on the Theory of Computing*, pages 793–802, 2013.

[29] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, Vol. 25 (2), pages 252–271, 1996.