On Testing Isomorphism to a Fixed Graph in the Bounded-Degree Graph Model*

Oded Goldreich and Laliv Tauber
Department of Computer Science
Weizmann Institute of Science, Rehovot, Israel.

September 28, 2023

Abstract

We consider the problem of testing isomorphism to a fixed graph in the bounded-degree graph model. Our main result is that, for almost all d-regular n-vertex graphs H, testing isomorphism to H 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.

The performance of our tester depends on natural graph parameters of the fixed (n-vertex) graph H such as its diameter and the minimum radius of "distinguishing neighborhoods" (i.e., the minimum $r = r(n)$ such that the "r-neighborhoods" of the n different vertices are pairwise non-isomorphic).

Contents

*Partially supported by the Israel Science Foundation (grant No. 1041/18) and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702).

1 Introduction

Following [6], we consider the problem of testing graph isomorphism in the bounded-degree graph model (introduced in [8] and reviewed in [5, Chap. 9])). Specifically, we consider the fixed graph version, where the input is a single (n-vertex) graph G, and the task is testing whether 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.

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).

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 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

[1] See [5, Sec. 12.7.2] for a brief discussion of "massively parametrized" properties.

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).

1.2 The most relevant previous results

The current work is most related to [6], which presented non-trivial lower bound on the complexity of testing isomorphism in the bounded-degree model. The focus of [6] was on the special case in which the graphs consist of small connected components; that is, n-vertex graphs with connected components of size $\mathrm{poly}(\log n)$. Ignoring the dependence on the proximity parameter, the main results presented in [6] are:

1. The query complexity of testing isomorphism to a fixed n-vertex graph (with connected components of size $\mathrm{poly}(\log n)$) is $\widetilde{\Theta}(n^{1/2})$.
2. The query complexity of testing isomorphism between two n-vertex graphs (with connected components of size $\mathrm{poly}(\log n)$) 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 [6] 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 [6], 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.

1.3 New results

Our results addresses the foregoing question. Specifically, we show that the lower bounds proved in [6] 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 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 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 [6, 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 [9].

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 1.1 asserts that such graphs, if they exist, are quite rare. Likewise, Theorem 1.2 asserts that fixed 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 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 1.4.1 (and is restated at the beginning of Section 2.1), and a generalization of it is captured by Definition 2.5. A condition that suffices for the lower-bound of Theorem 1.2 is stated in Definition 3.1.

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 2.6 and Corollary 3.4, resp.).

1.4.1 The tester: Proof sketch for Theorem 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 in Lemma 2.1, *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 connected 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) is not connected 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 connected 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 [10, 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] One can modify this example to obtain connected graphs by adding a single edge between the two $n/2$-vertex subgraphs.

1.4.2 The lower bound: Proof sketch for Theorem 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 [7, Def. 3.2.1]. 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 [7, Lem. 3.2.2], whereas the fact that random d-regular n-vertex graphs have $\mathtt{sc}$-value $O(1/n)$ is proved in [7, Lem. 3.2.3].

1.5 Other related work

The two versions of the graph isomorphism testing problem were considered before [4, 11, 12], in various models. Among these studies, the work of Newman and Sohler [12] is most relevant to us, since it is in the bounded-degree graph model; but, like [6] (which was reviewed in Section 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 [12, 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.

A few years earlier, both testing problems were studied by Fischer and Matsliah [4] *in the dense graph model* (reviewed in [5, 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 [11] *in the general graph model* (reviewed in [5, 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.

1.6 Organization

Section 2 contains the proof of Theorem 1.1. We first prove that the unique neighborhoods condition holds in random regular graphs (see Section 2.1), and then provide a more detailed description of the tester and its analysis (see Section 2.2). It is fair to say that Section 2.1 is more technical and mathematical in nature, whereas Section 2.2 is more interesting from an algorithmic point of view.

Section 3 provides a proof of Theorem 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 [7, Sec. 3.2].

2 The tester: Proof of Theorem 1.1

As stated in Section 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). We first prove that this hypothesis does hold for almost all d-regular n-vertex graphs.

2.1 The unique neighborhoods condition

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.

Lemma 2.1 (unique neighborhoods in random graphs): *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.*

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 3.2 (combined with [7, Lem. 3.2.3]).

Proof Sketch: For any vertex v in any d-regular n-vertex graph, we consider ℓ^*-long walks that are associated with ℓ^*-long sequences of choices of edges to traverse at the current vertices along the walk (i.e., these are non-backtracking walks (cf. [1])). Thus, the number of such walks is $m = d \cdot (d-1)^{\ell^*-1} = \widetilde{O}(n^{1/2})$. The probability that, *in a random d-regular n-vertex graph*, two such walks meet at their last step (i.e., *collide*), is approximately $1/n$, and so the expected number of pairwise collisions among all possible m walks is approximately $\binom{m}{2} \cdot (1/n) = \widetilde{O}(n^{1/2})^2/n = \mathrm{poly}(\log n)$. Furthermore, the actual number of collisions (in a random d-regular n-vertex graph) ranges mostly over $\mathrm{poly}(\log n)$ values. Hence, starting from different vertices, the corresponding m walks are likely to contain a different number of collisions. Alas, these numbers range in a set that is not large enough to distinguish all vertex-pairs. What we shall do instead is consider collisions among pairs of walks that have the same $(\log_{d-1}(m/n^{1/2}))$-long prefix.

Letting $\ell' = \log_{d-1}(m/n^{1/2}) = O(\log \log n)$, the point is that clustering the m walks according to their ℓ'-long prefix yields $d \cdot (d-1)^{\ell'-1} = \mathrm{poly}(\log n)$ random variables such that each of these random variables describes the existence of a collision among the corresponding $\Theta(n^{1/2})$ walks. Hence, each of these random variables is "quite random" (i.e., it is 1 with probability that is bounded away from both 0 and 1), and the $\mathrm{poly}(\log n)$ random variables are "sufficiently independent" so that there is hope to unique identify each vertex in a random graph according to the value of these random variables. Needless to say, the foregoing is a very rough sketch of our proof strategy, which will be clarified below.

(Indeed, our proof takes a middle path between two natural extremes. One extreme is to use the actual definition of an ℓ^*-neighborhood and consider, for each vertex u, the subgraphs induced by the set of vertices that are at distance at most ℓ^* from u in the graph. The other extreme is to consider, for each vertex u, only the number of different vertices that are at distance at most (or exactly) ℓ^* from u in the graph (equiv., the pattern of collisions). The middle path we take is to

consider, for each vertex u, the $k = \text{poly}(\log n)$ numbers that record the number of vertices in the $(\log_{d-1} n^{1/2})$-neighborhood of each vertex that is at distance $\log_{d-1} k$ from u.)

We highlight a key fact about our setting of parameters: Using $t \stackrel{\text{def}}{=} \lfloor \log_{d-1} n^{1/2} \rfloor - 1$ implies that $d \cdot (d-1)^{t-1} = \Theta(n^{1/2})$, where $d \cdot (d-1)^{i-1}$ represent the maximum number of vertices that are at distance exactly i from a given vertex. The point is the probability of a collision among $\Theta(n^{1/2})$ elements drawn uniformly and almost independently in $[n]$ is a constant in $(0,1)$. Intuitively, the same holds for the event of a collision (among the foregoing walks) in the t-neighborhood of a fixed vertex in a random graph. (Note that the existence of such a collision means that this t-neighborhood has less than $d \cdot (d-1)^{t-1}$ vertices.) Using $\ell' = O(\log \log n)$ implies that $k \stackrel{\text{def}}{=} d \cdot (d-1)^{\ell'-1} = \text{poly}(\log n)$, whereas a sequence of k almost independent 0-1 random variables that are each 1 with probability $p \in (0,1)$ yields a distribution over $\{0,1\}^k$ in which each value occurs with probability $\exp(-\Omega(k)) \ll 1/\text{poly}(n)$. Intuitively, this suggests that the sequence of the sizes of the t-neighborhoods of the vertices that are a t distance exactly ℓ' from a vertex in a random graph uniquely identifies this vertex. (Indeed, this intuition is inaccurate because the vertices in an ℓ'-neighborhood are not "ordered" but rather reside in leaves of an unlabeled tree; we shall capitalized on the tree structure by considering the multi-set of partial sums that are associated with vertices at distance $\ell'/3$ from a vertex of interest.)

The actual random variables. Let X denote a random d-regular graph over the vertex set $[n]$. The basic intuition is that, for $t = \lfloor \log_{d-1} n^{1/2} \rfloor - 1$ and $v \in [n]$, the $(t-\omega(1))$-neighborhood of v in X is likely to be a tree, whereas its $(t+\omega(1))$-neighborhood is likely to contain cycles. Furthermore, the number of vertices in the t-neighborhood of v is a random variable, denoted X_v, with a small but positive variance. Indeed, the number vertices is a t-neighborhood equals $1 + \sum_{i \in [t]} d \cdot (d-1)^{i-1}$ if and only if the corresponding t-long walks contain no collision. Now, the idea is to *uniquely identify each vertex u in X by the values of the X_v's for all v's that are at distance exactly $\ell' = O(\log \log n)$ from u*; typically, these v's will be the $d \cdot (d-1)^{\ell'-1}$ leaves of a (depth ℓ') tree rooted at u.

We stress that we will use not only the sum of these X_v's, but rather the "pattern" seen on this *marked* (depth ℓ') tree, where the leaf v is marked by the value of X_v. Actually, it suffices to consider the multi-set of the values that correspond to the different vertices w that are at distance $\ell'/3$ from u, where the value associated with vertex w is the sum of the X_v's that correspond to leaves of the (depth $2\ell'/3$) sub-tree rooted at w (see illustration in Figure 1, where $\ell'' = \ell'/3$).

For sake of simplicity, we shall actually modify the foregoing proposal (slightly). For every $\ell \in \mathbb{N}$ and $v \in [n]$, we let Ξ_v^ℓ be a random variable representing the set of vertices that are at distance exactly ℓ from the vertex v in the random graph X. For every $u \in [n]$ and $v \in \Xi_u^{\ell'}$, we define $\chi_{u,v} = 1$ if $|\Xi_u^{t+\ell'} \cap \Xi_v^t| < (d-1)^t$ and $\chi_{u,v} = 0$ otherwise; that is, *the binary random variable $\chi_{u,v}$ indicates whether there are collisions among the $(t+\ell')$-step walks from u to $\Xi_u^{t+\ell'}$ that go via v.*

Overview of the analysis. For starters, we claim that the probability of collision among these $(d-1)^t \approx \sqrt{n}$ walks is a constant in $(0,1)$; that is, $p \stackrel{\text{def}}{=} \Pr[\chi_{u,v} = 1] \in [\alpha_d, \beta_d]$, where $0 < \alpha_d < \beta_d < 1$ are constants. (The intuition here is that the endpoints of these $(d-1)^t$ random walks are almost uniformly and independently distributed in $[n]$.) Now, for $\ell' = 3\ell''$, we consider the random multi-set

$$\xi_u \stackrel{\text{def}}{=} \left\{ \sum_{v \in (\Xi_u^{3\ell''} \cap \Xi_w^{2\ell''})} \chi_{u,v} : \quad w \in \Xi_u^{\ell''} \right\} \tag{1}$$

where each element in the multi-set corresponds to a vertex w at distance ℓ'' from u, and represents the sum of the indicator variables that reside in vertices that are at distance $3\ell''$ from u and distance $2\ell'$ from w. Intuitively, each of the $|\Xi_u^{\ell''}|$ elements in the multi-set is distributed approximately as the binomial distribution of $(d-1)^{2\ell''}$ trials with success probability p, which implies that each possible value occurs with probability at most $\exp(-\Omega(\ell'')) = 1/\mathrm{poly}(\log n)$. Further assuming that these elements are distributed almost independently of one another, it follows that $\Pr[\xi_u = S] = \exp(-\omega(\log n))$ for every multi-set $S \in [[(d-1)^{2\ell''}]]^{|\Xi_u^{\ell''}|}$. Lastly, assuming that the ξ_u's are almost pairwise independent, it follows that $\Pr[\xi_{u_1} \neq \xi_{u_2}] = o(1/n^2)$ for every $u_1 \neq u_2$ in $[n]$.

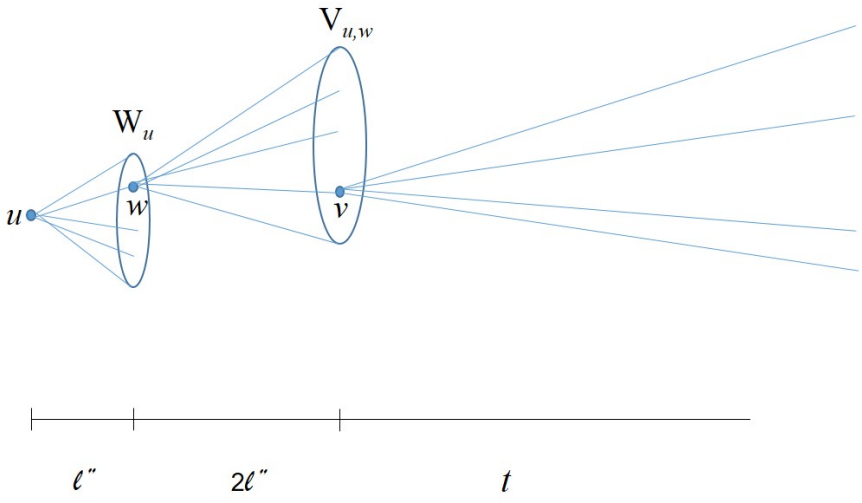

Figure 1: The $3\ell''$-neighborhood of u and the t-step walk from $v \in V_{u,w}$.

The actual analysis. Needless to say, we need to turn the foregoing hand-waving argument into a rigorous proof, where the main issue is that the various random variables are not fully independent. Nevertheless, these random variables are sufficiently random for the argument to goes through. Specifically, it suffices to prove the last assertion; that is, *for every $u_1 \neq u_2$ in $[n]$, it holds that* $\Pr[\xi_{u_1} \neq \xi_{u_2}] = o(1/n^2)$. Our first step towards this end is to explicitly define the random variables that make-up the random multi-set ξ_u of Eq. (1). Specifically, for every $u, w \in [n]$, we define the random variable

$$\xi_{u,w} \stackrel{\text{def}}{=} \sum_{v \in (\Xi_u^{3\ell''} \cap \Xi_w^{2\ell''})} \chi_{u,v} \tag{2}$$

and consider its distribution conditioned on $w \in \Xi_u^{\ell''}$. Such a conditioning can be enforced by fixing an ℓ''-long path from u to w. We shall actually fix the entire $3\ell''$-neighborhood of u, and let W_u denote the set of vertices at distance ℓ'' from u and $V_{u,w}$ denote the set of vertices that are at distance $3\ell''$ from u and distance ℓ'' from $w \in W_u$ (see illustration in Figure 1).

Let us first assume, for simplicity, that the $3\ell''$-neighborhood of u is a tree. In such a case, each t-long (non-backtracking) walk going out of $v \in \bigcup_{w \in W_u} V_{u,w}$ has $d-1$ options for each step, where we always avoid going back on the edge traversed in the previous step. Note that $\chi_{u,v} = 0$ if and only if all these $(d-1)^t$ possible t-step walk reach new and distinct vertices; that is, (E1) none of these walks hits the $3\ell''$-neighborhood of u, and (E2) these walks are vertex disjoint (except,

of course, at v). Letting Γ_u denote the $3\ell''$-neighborhood of u, the probability of event (E1) is lower-bounded by

$$\begin{aligned} 1-2\cdot(d-1)^t\cdot \Pr_{r\in[n]}[r\in\Gamma_u] &> 1-(d-1)^t\cdot\frac{d^{3\ell''}}{n} \\ &= 1-\mathrm{poly}(\log n)\cdot n^{-1/2} \;=\; 1-o(n^{-1/3}), \end{aligned}$$

where $\sum_{i=0}^{t}(d-1)^i < 2\cdot(d-1)^t$ is the number of vertices on all t-step walks from vertex v, and $d^{3\ell''}$ in an upper bound on the number of vertices in the $3\ell''$-neighborhood of v. (We stress that the next vertex on a walk is not selected uniformly among all vertices, but is rather selected in proportion to the unused incidences; hence, this choice is actually slanted against vertices that were already used.) Likewise, the probability of event (E2) is lower-bounded by

$$1-\binom{2\cdot(d-1)^t}{2}\cdot\frac{1}{n} > 1-2\cdot\frac{(d-1)^{2t}}{n} > 1-2\cdot(d-1)^{-2},$$

since $t\leq(\log_{d-1}n^{1/2})-1$. Hence, $prob[\chi_{u,v}=0] > 1-(2\cdot(d-1)^{-2}+o(n^{-1/3}))$. On the other hand, as shown next, the probability of event (E2) is upper-bounded by $1-0.1\cdot(d-1)^{-4}$. It follows $p_d \stackrel{\rm def}{=} \Pr[\chi_{u,v}=1]$ resides in the bounded interval $[0.1\cdot(d-1)^{-4}\,,\,2\cdot(d-1)^{-2}+o(n^{-1/3})]$.

Claim: *For u and v as above,* $\Pr[\chi_{u,v}=1]\geq 0.1\cdot(d-1)^{-4}$.

Proof: For every pair of distinct (non-backtracking) walks $\overline{\alpha},\overline{\beta}\in[d-1]^t$, we let $\chi_{u,v}^{(\overline{\alpha},\overline{\beta})}$ be a 0-1 random variable indicating the event that the endpoint of the walk $\overline{\alpha}$ (from v) collides with the endpoint of the walk $\overline{\beta}$. Letting $\mathcal{P}\stackrel{\rm def}{=}\binom{[d-1]^t}{2}$ denote the set of pairs of walks, we have

$$\begin{aligned} \Pr[\chi_{u,v}=1] &\geq \Pr\left[\sum_{\{\overline{\alpha},\overline{\beta}\}\in\mathcal{P}}\chi_{u,v}^{(\overline{\alpha},\overline{\beta})}>0\right] \\ &\geq \sum_{\{\overline{\alpha},\overline{\beta}\}\in\mathcal{P}}\mathrm{E}\left[\chi_{u,v}^{(\overline{\alpha},\overline{\beta})}\right] - \sum_{\{\{\overline{\alpha},\overline{\beta}\},\{\overline{\alpha}',\overline{\beta}'\}\}\in\binom{\mathcal{P}}{2}}\mathrm{E}\left[\chi_{u,v}^{(\overline{\alpha},\overline{\beta})}\cdot\chi_{u,v}^{(\overline{\alpha}',\overline{\beta}')}\right] \end{aligned} \tag{3}$$

where the first inequality discards the probability of event (E1). Recalling that $\Pr[\chi_{u,v}^{(\overline{\alpha},\overline{\beta})}=1]<1/n$, observe that $\Pr[\chi_{u,v}^{(\overline{\alpha},\overline{\beta})}=1]>(d-1)/dn>1/2n$, because the endpoint of the first walk (i.e., $\overline{\alpha}$) has $d-1$ unused ports, which may each be chosen as the endpoint of the second walk (i.e., $\overline{\beta}$). Also note that $\Pr[\chi_{u,v}^{(\overline{\alpha}',\overline{\beta}')}=1|\chi_{u,v}^{(\overline{\alpha},\overline{\beta})}=1]<1/n$. Using these three bounds, we lower-bound Eq. (3) by

$$|\mathcal{P}|\cdot\frac{1}{2n}-|\mathcal{P}|^2\cdot\frac{1}{n^2} \;=\; \frac{|\mathcal{P}|}{n}\cdot\left(0.5-\frac{|\mathcal{P}|}{n}\right).$$

Recalling that $|\mathcal{P}|=\binom{(d-1)^t}{2}\approx(d-1)^{2t}/2$ and using $|\mathcal{P}|>0.4\cdot(d-1)^{2t}>0.4\cdot n\cdot(d-1)^{-4}$ and $|\mathcal{P}|<0.5\cdot(d-1)^{2t}\leq 0.5\cdot n\cdot(d-1)^{-2}$, we infer that

$$\Pr[\chi_{u,v}=1] \;>\; 0.4\cdot(d-1)^{-4}\cdot\left(0.5-0.5\cdot(d-1)^{-2}\right) \;\geq\; 0.4\cdot(d-1)^{-4}\cdot\frac{3}{8}$$

where here (and elsewhere) we used $d-1\geq 2$. The claim follows. ■

Recap and beyond. So far we analyzed the distribution of t-step random walks from a vertex $v \in V_{u,w}$, where $v \in W_w$, conditioned on a fixing of the $3\ell''$-neighborhood of u (which in turn determines W_u and $V_{u,w}$). A similar analysis holds also when fixing $(d-1)^t$ possible t-step walks for each other member of $\bigcup_{w' \in W_u} V_{u,w'}$, except that we need to consider a third event – the possibility that some of the t-step walks from v hit some t-step walks from other vertices of $\bigcup_{w' \in W_u} V_{u,w'}$.

The latter event is not rare (since there are $(d-1)^t$ walks from v and $\Theta((d-1)^{3\ell''})$ times more walks from the other vertices in $\bigcup_{w' \in W_u} V_{u,w'}$), but we can ignore its effect on the collisions between walks from v. Specifically, we may assume that the t-step walks from v that hit t-step walks from vertices of $\bigcup_{w' \in W_u} V_{u,w'} \setminus \{v\}$ do not hit other t-step walks from v. This is the case because, with very high probability (i.e., probability $1 - o(1/n^2)$), the number of t-step walks from v that intersect t-step walks from other vertices v' of $\bigcup_{w' \in W_u} V_{u,w'}$ is relatively small (i.e., $\mathrm{poly}(\log n)$). Hence, even under the foregoing conditioning (i.e., conditioning on all other $\chi_{u,v'}$'s), it holds that $\Pr[\chi_{u,v}=1] = p_d \pm o(n^{-1/3})$.

It follows that the different $\xi_{u,w}$'s are each the sum of $|V_{u,w}|$ binary random variables (i.e., all $\chi_{u,v'}$ for $v' \in V_{u,w}$), that are each 1 with probability $p_d \pm o(1)$, also when conditioned on all other random variables associated with $\bigcup_{w' \in W_u} V_{u,w'}$. Thus, for every $x \in \mathbb{N} \cup \{0\}$ and $\overline{x} \in (\mathbb{N} \cup \{0\})^{|W_u|-1}$, it holds that

$$\Pr\left[\xi_{u,w}=x \,\middle|\, (\xi_{u,w'})_{w' \in W_u \setminus \{w\}} = \overline{x}\right] \;=\; O(|V_{u,w}|^{-1/2}) \;=\; O((d-1)^{-\ell''})$$

because $|V_{u,w}| = (d-1)^{2\ell''}$. Hence, $\xi_{u,w}$ is distributed over $q \stackrel{\text{def}}{=} \Omega((d-1)^{\ell''})$ values such that no value has probability higher than $1/q$. Recalling that ξ_u denotes the multi-set $\{\xi_{u,w} : w \in W_u\}$, it follows that, for every multi-set S of $\mathbb{N} \cup \{0\}$, it holds that

$$\Pr[\xi_u = S] \;\leq\; \frac{1}{\binom{q+|W_u|-1}{|W_u|-1}} \;=\; \exp(-\Omega((d-1)^{\ell''})), \tag{4}$$

because $|W_u| = \Omega((d-1)^{\ell''})$.

The foregoing analysis was conducted while assuming that the $3\ell''$-neighborhood of u is a tree (equiv., $|W_u| = d \cdot (d-1)^{\ell''-1}$ and $|V_{u,w}| = (d-1)^{2\ell''}$ for every $w \in W_u$), but it actually holds as long as $|W_u| = \Omega((d-1)^{\ell''})$ and $|V_{u,w}| = \Omega((d-1)^{3\ell''})$ for every $w \in W_u$. The latter condition holds with probability $1 - o(1/n^2)$; that is, $\Pr[|\Xi_u| = \Omega((d-1)^{\ell''}] = 1 - o(1/n^2)$ and $\Pr[(\forall w \in \Xi_u^{\ell''}) \, |\Xi_u^{3\ell''} \cap \Xi_w^{2\ell''}| = \Omega((d-1)^{2\ell''}] \;=\; 1 - o(1/n^2)$.

Lastly, we note that all of the forgoing holds also when fixing the $(t+3\ell'')$-neighborhood of any other vertex u', provided that this fixing satisfies the foregoing conditions (on the sizes of $\Xi_{u'}^{\ell''}$ and of $\Xi_{u'}^{3\ell''} \cap \Xi_w^{2\ell''}$ for every $w \in \Xi_u^{\ell''}$). In this case, when considering the t-step walks from $v \in \Xi_u^{3\ell'}$, we need to ignore also the t-step walks that hit t-step walks from vertices in $\Xi_{u'}^{3\ell'}$, rather than only the t-step walks from the other vertices in $\Xi_u^{3\ell'}$. We can afford ignoring these $\widetilde{O}(n^{1/2})$ walks, just as we ignored the former $\widetilde{O}(n^{1/2})$ walks. It follows that $\Pr[\xi_u = \xi_{u'}] = o(1/n^2)$ for every $u \neq u'$ in $[n]$. Note that the $o(1/n^2)$ term is due to various bad events that we set aside, which dominate the $\exp(-\Omega((d-1)^{\ell''}))$ term that is due to Eq. (4) (provided that $\ell'' \geq \log_{d-1} O(\log n)$). ∎

2.2 The tester

For any fixed graph H of logarithmic diameter that satisfies the unique neighborhoods condition, our tester proceeds as outlined in Section 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 (locating vertices of a fixed graph in an input graph): *Suppose that H is a fixed d-regular n-vertex 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 is $D \cdot d \cdot (d \cdot (d-1)^{\ell^*-1}) = O(D \cdot (d-1)^{\ell^*})$, which is $\mathrm{poly}(\log n) \cdot n^{1/2}$ when $\ell^* = \log_{d-1} \widetilde{O}(n^{1/2})$ and $D = \mathrm{poly}(\log n)$. Using Algorithm 2.2, we finally detail our tester.

Algorithm 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. 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 2.3 has query complexity $O(\epsilon^{-1} \cdot (d-1)^{\ell^*} \cdot D)$, which is $\mathrm{poly}(\log n) \cdot n^{1/2}/\epsilon$. 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 2.4 (on the graphs accepted by Algorithm 2.3): *If the Algorithm 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 does not fail. For each $u \in U$, let $\mu(u)$ denote the location of u in G as determined by Algorithm 2.2, and note that μ is a 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 1.1. Recall that Lemma 2.1 asserts that almost all d-regular n-vertex graphs satisfy the unique ℓ^*-neighborhoods condition, and ditto regarding having logarithmic diameter. Using Algorithm 2.3 (as analyzed in Claim 2.4), we establish Theorem 1.1.

Abstraction and generalization. As stated in Section 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) 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 [10, Sec. 4.4]. We review it next (following the more general formalism of [7, Def. 1.2]).

Definition 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 can be

[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.

[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).

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 [10, Thm. 4.9]. Plugging this algorithm in Algorithm 2.3, we get the following result.

Theorem 2.6 (generalization of Theorem 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 2.3, we replace the exploration of ℓ^*-neighborhoods used inside Algorithm 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. At this point, the analysis proceeds as in the proof of Claim 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 (instead of the exploration of ℓ^*-neighborhoods). Hence, when Algorithm 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 2.3 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 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 2.3 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. ■

3 Proof of the lower bound (Theorem 1.2)

Theorem 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

$$\left|\Pr_{G\sim\mathcal{D}_1}[A_H^G=1] - \Pr_{G\sim\mathcal{D}_2}[A_H^G=1]\right| = o(1). \tag{5}$$

   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 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 [7, 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 event of type (2) occurs with probability $O(q^2/n)$, and the focus of [7, Lem. 3.2.2] is on showing that the same bound holds for events of type (1).

[5] The latter claim follows by a straightforward counting argument. Recalling that the number of labeled d-regular n-vertex graphs [2, 3] is $N_d(n) \stackrel{\text{def}}{=} \Theta((dn/e)^{dn/2}/(d!)^n)$, 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 $M_{d,\epsilon}(n) \stackrel{\text{def}}{=} n! \cdot \binom{dn/2}{\epsilon\cdot dn/2} \cdot n^{\epsilon\cdot dn/2}$. Hence, $M_{d,\epsilon}(n)/N_d(n)$ is upper-bounded by $(d/n)^{dn/2} \cdot n^{(1+0.5\epsilon d)\cdot n} \cdot 2^{H_2(\epsilon)dn/2}$, 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.

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 [7, 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 [7, Sec. 3.2], we consider the probability that a non-backtracking random walk consists of a simple cycle.

Definition 3.1 (probability of forming a simple cycle [7, 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 [7, 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 [7, Lem. 3.2.2].

Lemma 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 [7, Lem. 3.2.2] is devoted to proving Lemma 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 [7, 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 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 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. (5)*.*

Corollary 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 3.3 with [7, 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 1.2. More generally, we get.

[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.

Corollary 3.4 (generalization of Theorem 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 $\Omega(\min(\mathtt{sc}(H)^{-1/2}, \mathtt{sc}(K)^{-1/2}, n^{1/2}))$.*

References

[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] 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.

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

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

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

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

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

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

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

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

[11] 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.

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