Lecture Notes on Testing Graph Properties in the General Graph Model

Oded Goldreich[*]

May 9, 2016

Summary: This lecture is devoted to testing graph properties in the general graph model, where graphs are inspected via incidence and adjacency queries, and distances between graphs are normalized by their actual size (i.e., actual number of edges). The highlights of this lecture include:

1. Demonstrating the derivation of testers for this model based on testers for the bounded-degree graph model.
2. Studying the tasks of estimating the average number of edges in a graph and sampling edges uniformly at random.

We concluded this chapter with some reflections regarding the three models of testing graph properties.

The current notes are based on several sources; see Section 5.2 for details.

Teaching note: Although it is possible to study the current chapter without first studying the Chapter 9, we strongly recommend not doing so. Basic familiarity with the bounded-degree graph model seems necessary for a good perspective on the general graph model. In addition, familarity with some of the results and ideas of Chapter 9 will greatly facilitate the study of current chapter. Specifically, it will be most beneficial to be familiar with the connectivity tester and the bipartiteness tester.

Organization. Following an introduction to the general graph model (Section 1), we study the issues that arise when trying to extend tester for the bounded-degree graph model to testers for the current model (Section 2). Next, in Section 3, we study the related problems of estimating the average degree in a general graph and selecting random edges in it, presenting two different algorithmic approaches towards solving these problems (see Sections 3.2.1 and 3.2.2, respectively). As illustrated in Section 2.2, these problems are pivotal for the design of some testers. Lastly, in Section 4, we illustrate the possible benefits of using both incidence and adjacency queries.

Teaching note: Again, we recommend covering only part of the material in class, and leaving the rest for optional independent reading. Aside from Section 1, which seems a must, the choice of what to teach and what to leave out is less clear. If pressed for our own choice, then the fact is that we chose to cover Sections 2.2 and 3.2.1.

[*]Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel.

1 The General Graph Model: Definitions and issues

The general graph model is intended to capture arbitrary graphs, which may be neither dense nor of bounded-degree. Such graphs occur most naturally in many settings, but they are not captured (or not captured well) by the models presented in the last two lectures (i.e., the dense graph model and the bounded-degree graph model).

Recall that both in the dense graph model and in the bounded-degree graph model, the query types (i.e., ways of probing the tested graph) and the distance measure (i.e., distance between graphs) were linked to the representation of graphs as functions. In contrast to these two models, in the general graph model the representation is blurred, and the query types and distance measure are decoupled.

Giving up on the representation as a yardstick for the relative distance between graphs, leaves us with no absolute point of reference. Instead, we just define the relative distance between graphs in relation to the actual number of edges in these graphs; specifically, the relative distance between the graphs $G = ([k], E)$ and $G' = ([k], E')$ may be defined as $\frac{|E \triangle E'|}{\max(|E|,|E'|)}$, where $E \triangle E' = (E \setminus E') \cup (E' \setminus E)$ denotes the symmetric difference between E and E'. Indeed, the normalization by $\max(|E|, |E'|)$ is somewhat arbitrary, and alternatives that seem as natural include $|E| + |E'|$, $|E \cup E'|$ and $(|E| + |E'|)/2$; yet, all these alternatives are within a factor of two from one another (and the slackness is even smaller in the typical case where $|E \cup E'| \approx |E \cap E'|$).

Turning to the question of query types, we again need to make a choice, which is now free from representational considerations. The most natural choice is to allow both *incidence queries* and *adjacency queries*; that is, we allow the two types of queries that were each allowed in one of the two previous models. Hence, the graph $G = ([k], E)$ is (redundantly) represented by (or rather accessed via) two functions:

1. An incidence function $g_1 : [k] \times [k-1] \to \{0, 1, ..., k\}$ such that $g_1(u, i) = 0$ if u has less than i neighbors and $g_1(u, i) = v$ if v is the i^{th} neighbor of u. That is, if $d_G(u)$ denotes the degree of u in G, then $\{g_1(u, i) : i \in [d_G(u)]\} = \{v : \{u, v\} \in E\}$.

   Indeed, here $k - 1$ serves as a (trivial) degree bound. (Recall that the bounded-degree graph model relied on an explicit degree bound, which was denoted d.)

2. An adjacency predicate $g_2 : [k] \times [k] \to \{0, 1\}$ such that $g_2(u, v) = 1$ if and only if $\{u, v\} \in E$.

Typically, adjacency queries become more useful when the graph becomes more dense, whereas incidence queries (a.k.a neighbor queries) become more useful when the graph becomes more sparse (cf. [3]). Nevertheless, both types of queries are allowed in the current model, and at times both are useful (see, e.g., Algorithm 13).

1.1 Perspective: Comparison to the two previous models

Recall that in the bounded-degree graph model we have implicitly assumed that the degree bound, denoted d, is of the same order of magnitude as the actual average degree (i.e., $d = O(|E|/k)$, or, equivalently, $|E| = \Omega(dk)$). This assumption is mute in case d is viewed as a constant, but the meaningfulness of the model for the case of varying d relies on the assumption that the average degree is $\Omega(d)$, or so. When this assumption holds, the difference between the measure of relative distance used here (i.e., in the general graph model) and the measure used in the bounded-degree graph model is not significant.

Likewise, in the dense graph model we have implicitly assumed that the density of edges is a constant (i.e., $|E| = \Omega(k^2)$). Whenever this assumption holds, the difference between the measure of relative distance used here (i.e., in the general graph model) and the measure used in the dense graph model is not significant.

We also note that in each of the two previous models, when the corresponding implicit assumption holds, it was easy to approximate the number of edges in the graph and to sample an edge uniformly at random: In the bounded-degree graph model, if $|E| = \Omega(dk)$, then we can approximate $|E|$ (resp., select an edge at random) by uniformly selecting $(u, i) \in [k] \times [d]$ at random, and checking whether $g_1(u, i) \in [k]$ (resp., output $\{u, g_1(u, i)\}$ if and only if $g_1(u, i) \in [k]$). Hence, we obtain a $1 \pm \epsilon$ factor approximation by repeating this experiment for $O(1/\epsilon^2)$ times (resp., obtain a random edge after $O(1)$ trials).[1] Likewise, in the dense graph model, if $|E| = \Omega(k^2)$, then we can approximate $|E|$ (resp., select an edge at random) by uniformly selecting $(u, v) \in [k] \times [k]$ at random, and checking whether $g_2(u, v) = 1$ (resp., output $\{u, v\}$ if and only if $g(u, v) = 1$). Hence, we obtain a $1 \pm \epsilon$ factor approximation by repeating this experiment for $O(1/\epsilon^2)$ times (resp., obtain a random edge after $O(1)$ trials).[2]

1.2 The actual definition

For sake of good order, we explicitly present the definition of testing graph properties in the general graph model.

Definition 1 (testing graph properties in the general model): *A* tester *for a graph property Π is a probabilistic oracle machine that, on input parameters k and ϵ and access to functions answering incidence queries and adjacency queries regarding an k-vertex graph $G = ([k], E)$, 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 k-vertex graph $G' = ([k], E') \in \Pi$ it holds that the symmetric difference between E and E' has cardinality that is greater than $\epsilon \cdot \max(|E|, |E'|)$.*

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. *A tester is called* non-adaptive *if it determines all its queries based solely on its internal coin tosses* (and the parameters k and ϵ)*; otherwise, it is called* adaptive.

The query complexity of a tester is the total number of queries it makes to any graph $G = ([k], E)$, as a function of the graph's parameters (i.e., k and $|E|$) and the proximity parameter ϵ. We stress that we count both the incidence queries and the adjacency queries, and each of these queries is counted as one unit.

As stated upfront, the motivation for the model captured by Definition 1 is to allow the consideration of arbitrary graphs (which may be neither dense nor of bounded-degree). In doing so, this model strengthens the relation between property testing and standard algorithmic studies. On the other hand, forsaking the paradigm of representing graphs as functions means that the connection to the rest of property testing is a bit weakened (or at least becomes more cumbersome).

[1]In both cases, the O-notation hides a factor of $dk/|E|$.
[2]In both cases, the O-notation hides a factor of $k^2/|E|$.

On extremely sparse graphs. Extremely sparse graphs, in which the number of edges is significantly smaller than the number of vertices, raise conceptual questions regarding the testing model captured by Definition 1. Conceptually, defining the relative distance between graphs as a fraction of the number of edges in these graphs represents the feeling that the number of edges in a graph represent its size. This is indeed the case in the typical cases in which the graph is not extremely sparse (i.e., the number of edges in the graph is at least of the same order of magnitude as the number of vertices). But in the pathological case of extremely sparse graphs, it seems that the number of vertices represents its size better. Hence, in general, it seems that $k + |E|$ represents the size of the graph $G = ([k], E)$ better than either $|E|$ or k. This leads to the following revision of the testing model (where the difference is at the very end of the definition).

Definition 2 (testing graph properties in the general model, revised): *A tester in the revised model is defined as in Definition 1, except that the definition of distance to the graph property* Π *is modified so that the graph* $G = ([k], E)$ *is said to be* ϵ-far *from* Π *if for every* k*-vertex graph* $G' = ([k], E') \in \Pi$ *it holds that the symmetric difference between* E *and* E' *has cardinality that is greater than* $\epsilon \cdot (k + \max(|E|, |E'|))$.

Hence, Definitions 1 and 2 differ only in that in Definition 1 the symmetric difference is divided by $\max(|E|, |E'|)$, whereas in in Definition 2 the symmetric difference is divided by $k + \max(|E|, |E'|)$. This difference is insignificant whenever $|E| = \Omega(k)$; that is, for $|E| = \Omega(k)$, the definitions of distance underlying Definitions 1 and 2 coincide up to a constant factor, which we ignore just as we ignored the difference between $|E| + |E'|$, $|E \cup E'|$, $\max(|E|, |E'|)$ and $(|E| + |E'|)/2$. Furthermore, the definitions collide (up to a constant factor) also in case the property contains only graphs $G' = ([k], E')$ such that $|E'| = \Omega(k)$ (e.g., `Connectivity`).

We note that the two definitions do differ when applied to properties that contain the empty graph (or any other extremely sparse graphs). Consider, for example, the case of `Bipartitness`. In this case, the k-vertex graph that consists of a single triangle and $k - 3$ isolated vertices is deemed 0.33-far from `Bipartiteness` by Definition 1, but Definition 2 views it as $1/k$-close to `Bipartiteness`. Hence, $\Omega(1)$-testing `Bipartiteness` under Definition 1 requires $\Omega(k)$ queries, merely due to the need to find a tiny portion of the graph that violates the property. But this need stands in contrast to the entire mindset of property testing that postulates that small parts of the object that violate the property can be ignored. The source of trouble is that Definition 1 may view such small portions as a large fraction of the object. We believe that the foregoing illustrates our opinion that Definition 1 does not account properly for the "size" of the tested object (when the tested object is an extremely sparse graph).

In light of the above, we believe that Definition 2 should be preferred over Definition 1. Hence, whenever there is a significant difference between these two definitions, we shall use Definition 2.

2 On obtaining testers for the current model

The general graph model is closer in spirit to the bounded-degree graph model than to the dense graph model, since the focus of the two former models is on sparse (or at least non-dense) graphs. The main difference between the general graph model and the bounded-degree model is that the former deals with graphs in which *vertex degree may vary in an extreme manner*. An additional issue is that the dependence of the complexity on the average vertex degree is viewed as more important. (We shall elaborate on these two issues shortly.)

Since a tester in the general graph model must definitely work also in the bounded-degree graph model and since designing testers in the general graph model is typically more difficulty, it makes sense to try first to design a tester for the bounded-degree graph model. (One may object the foregoing assertion by claiming that the general graph model endows the tester with additional power (i.e., it allows adjacency queries), but this power is irrelevant in the case that the graph is very sparse.)[3]

In light of the foregoing, designing testers for the general graph model may be viewed as adapting and/or extending testers designed for the bounded-degree graph model to the general graph model. Such an adaptation and/or extension faces the two aforementioned difficulties (or issues).

1. *The dependence of the original tester on the degree bound*: In the bounded-degree graph model, one tends to ignore the dependence of the complexity of testing on the degree bound, denoted d, which is often viewed as a constant. Note that this parameter has two opposite effects. On the one hand, when d increases, relative distances decrease, and so testing may become easier. On the other hand, the complexity of some operations (e.g., scanning all neighbors of a given vertex) may grow with d. So the first challenge is figuring out the exact effect of d on the complexity of the original tester.

   For example, the tester for `Bipartiteness`, presented in [10], was originally analyzed assuming that the degree bound is a constant, which led to ignoring the dependence of its complexity on the degree bound. Fortunately, a closer look at the analysis, taken in [13], revealed that the complexity does not grow with the degree bound (since the two opposite effects cancel out).[4] Note that this is the case also with the tester for `connectivity` (see previous lecture).

2. *The effect of drastically varying vertex degrees*: A more acute problem with the bounded-degree graph model is that it tends to blur the difference between the average degree of the graph and its maximal degree. But in the general graph model these two quantities play different roles. The average degree is used to normalize the relative distance of the input graph to the property (since this distance is normalized by the input's average degree), but the query complexity may depend on the maximal degree. (In contrast, in the bounded-degree graph model, the relative distance is also normalized by the maximum degree.)

   Hence, when these two quantities are significantly different, the aforementioned cancelling effect does not apply, and typically we cannot use the tester for the bounded-degree graph model as is. Instead, we should adapt the tester so that its operation is better tailored to the varying degrees of the vertices of the input graph. There are two ways of doing so.

   (a) An explicit adaption: Changing the original tester, by possibly generalizing an idea that underlies the original design.

   (b) A reduction: The original tester remains intact, but it is not applied to the input graph but rather to a graph that is derived from it by a local reduction. Needless to say, in this case, the reduced graph will have a maximum degree that is of the same order of magnitude as its average degree.

   We shall demonstrate both ways next.

[3]Formally, adjacency queries can be emulated by d incidence queries, when d is the maximum degree in the graph. Hence, when d is a constant, we gain very little by using adjacency queries.

[4]This can be seen in the special case of rapid-mixing, which was analyzed in the previous lecture.

Before turning to the actual demonstrations, let us comment that the second approach (i.e., using a reduction) *seems* to require estimating the average degree of the graph (as well as selecting edges uniformly at random in the input graph). But as will be shown in Section 3.1, estimating the average degree of a k-vertex graph requires query complexity $\Omega(\sqrt{k})$, at least in case that it has $O(k)$ edges, and so this route is to be avoided when seeking lower complexity.

Teaching note: In Sections 2.1 and 2.2 we demonstrate the two routes outlined in Items 2a and 2b, respectively, while referring to the specific testers for `Connectivity` and `Bipartiteness`. Since our focus is on demonstrating the general principles, we do not provide detailed analyses of the two resulting testers, but rather confine ourselves to overviews.

2.1 An explicit adaptation: the case of connectivity

The tester for `Connectivity` in the bounded-degree graph model, presented in the previous lecture, can be easily adapted to the current context. We first recall that *if a graph has m connected components, then it can be made connected by adding $m-1$ edges.* In our context, this means that the k-vertex graph is m/k-close to being connected; specifically, if $G = ([k], E)$ has m connected components, then it is ϵ-close to being connected for $\epsilon = \frac{m-1}{|E|+m-1} \leq \frac{m-1}{k-1} \leq \frac{m}{k}$ (since $|E| + m - 1 \geq k - 1$ and $k \geq m$).[5]

Recall that the tester for `Connectivity` (in the bounded-degree graph model), presented in the previous lecture, consisted of conducting truncated BFSes that are suspended once a specified number of vertices, denoted s, is encountered. In the context of the bounded-degree graph model, the complexity was $d \cdot s$, where d was the degree bound. But in the current setting, there is no degree bound, and so the complexity of the search is bounded by s^2 (i.e., the maximum possible number of edges in the subgraph induced by these s vertices). Hence, the complexity of the tester is $\widetilde{O}(1/\epsilon^2)$ rather than $\widetilde{O}(1/\epsilon)$.

A minor improvement over the foregoing upper bound can be obtained by a different setting of the parameters in Levin's economical work investment strategy. Specifically, for $i = 0, 1, ..., \ell \stackrel{\text{def}}{=} \log(1/\epsilon)$, we select at random $O(i^2 \cdot 2^i)$ start vertices, and conduct a truncated BFS from each of them such that the search is suspended once $O(2^{-i}/\epsilon)$ vertices are encountered. Using an analysis as in the lecture on the dense graph model, we obtain:

Theorem 3 (testing connectivity (in the general graph model)):[6] *Connectivity has a* (one-sided error) *tester of time* (and query) *complexity* $O(1/\epsilon^2)$.

2.2 Using a reduction: the case of Bipartiteness

Our aim here is to present an extension of the testing result for `Bipartiteness` from the bounded-degree graph model to the general graph model.

Theorem 4 (testing `Bipatiteness` (in the general graph model)):[7] `Bipatiteness` *has a* (one-sided error) *tester of time* (and query) *complexity* $\text{poly}(1/\epsilon) \cdot \widetilde{O}(\sqrt{k})$.

[5]Recall that the bound used in the previous lecture was $\frac{2m-1}{dk/2}$, where d was the degree bound (and the factor of two was due to the need to preserve the degree bound).

[6]This result holds under both testing models presented in Section 1 (cf. Definitions 1 and 2).

[7]We stress that this result refers to the testing model captured by Definition 2.

This result will be established by a *reduction from testing in the general graph model to testing in the bounded-degree graph model*. That is, we shall transform the input graph, which may have vertices of significantly varying degree, into a graph that fits the bounded-degree graph model, and apply the original tester on the resulting graph. But first, we observe that the performance of the original tester (for `Bipartiteness` in the bounded-degree graph model) does not deteriorate when the degree bound d increases. Furthermore, its analysis continues to hold also if the input graph is not simple (i.e., has parallel edges). The reader can easily verify both claims for the special case of rapid-mixing, presented in the previous lecture. Hence, we focus on handling the case that the degree in the input graph $G = ([k], E)$ vary significantly; that is, the average degree $\bar{d} \overset{\text{def}}{=} 2|E|/k$ may be much smaller than the maximal degree $d_{\max} \overset{\text{def}}{=} \max_{v\in[k]}\{d_G(v)\}$, where $d_G(v) \overset{\text{def}}{=} |\{u : \{u,v\}\in E\}|$. For sake of simplicity, we assume that $\bar{d} \geq 1$ is an integer.

We obtain a tester for `Bipartiteness` in the general graph model by invoking the original tester on an imaginary graph that is obtained by replacing vertices of high degree in the input graph $G = ([k], E)$ with adequate *gadgets*, and distributing the edges incident at the original high-degree vertices among the vertices of these gadgets. Specifically, a vertex v having degree $d_G(v)$ is replaced by a $O(d_G(v)/\bar{d})$-vertex graph G_v of maximal degree $\bar{d}$, while connecting the original neighbors of v to vertices of the gadget G_v such that each vertex of the gadget has at most $\bar{d}$ *external edges* (which lead to other gadgets, which replace the neighbors of v).[8]

Needless to say, this intended replacement of vertices by gadgets should preserve the distance of the original graph from being bipartite (up to a constant factor). That is, if G is bipartite, then the resulting graph G' should be bipartite, and if G is ϵ-far from being bipartite, then G' should be $\Omega(\epsilon)$-far from being bipartite. In such a case, we obtain a (local) reduction of testing `Bipartiteness` in the general graph model to testing `Bipartiteness` in the bounded-degree graph model.

The choice of adequate gadgets is, of course, crucial. For starters, these gadgets should be bipartite graphs, since otherwise bipartiteness is not preserved by the replacement. Furthermore, for connections to other gadgets, we should only use vertices of one side of the bipartite graph, called its external side. However, preserving the distance to `Bipartitness` requires more than that (i.e., more than preserving the distance in case it is zero). We want it to be the case that if the original graph G is far from being bipartite, then so is the resulting graph G'; equivalently, if G' is close to being bipartite, then so is G. This will be the case if for every 2-partition of G' that has few violating edges (i.e., edges with both endpoints on the same side), the gadgets "force" placing all the external vertices of each gadget on the same side of the 2-partition. In such a case, the 2-partition of G' (with few violating edges) induces a 2-partition of G with a few violating edges.

(The reader may observe that the foregoing feature is satisfied by random $\bar{d}$-regular bipartite graphs (see Exercise 1), and may assume at this point that this is what we use, although we shall actually use $\bar{d}$-regular bipartite graphs that are expanders in a sense to be defined in the sequel.)

In any case, a vertex v having degree $d_G(v)$ is replaced by a $\bar{d}$-regular bipartite graph G_v with $t_v = \lceil d_G(v)/\bar{d}\rceil$ vertices on each side, while connecting the original neighbors of v (or rather the gadgets that replace them) to vertices on the external side of G_v. That is, at most $\bar{d}$ neighbors of v (or rather vertices in other gadgets) are connected to each vertex on the external side of the

[8]Indeed, it unnecessary to apply this replacement to vertices of degree at most $\bar{d}$, but it is simpler to apply it also to these vertices. In this case (i.e., when $d_G(v) \leq \bar{d}$), the graph G_v consists of a single pair of vertices that are connected by $\bar{d}$ parallel edges, which means that v remains connected to its original neighbors, but also gets connected to a new auxiliary neighbor.

bipartite gadget G_v, whereas the vertices on the "internal" side of G_v are only connected to the "external" vertices of G_v (see Figure 1).[9]

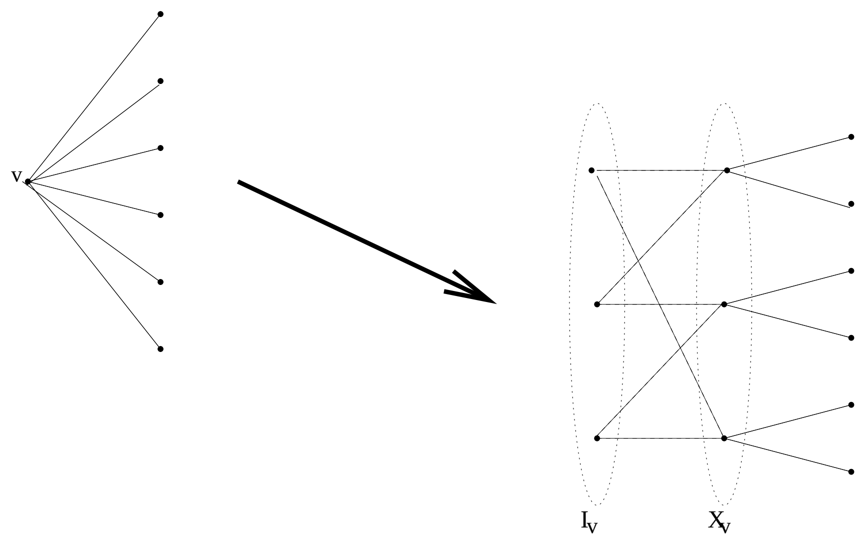

Figure 1: The figure depicts the case of $\bar{d} = 2$ and $t_v = 3$. The vertex v is replaced by a bipartite graph with sides X_v and I_v, where "X" stands for external and "I" for internal.

Hence, vertices in the resulting graph, denoted G', have degree at most $2\bar{d}$. Note that the number of vertices in G' is

$$\sum_{v\in[k]} 2 \cdot \lceil d_G(v)/\bar{d}\rceil \;<\; 2 \cdot \sum_{v\in[k]} ((d_G(v)/\bar{d}) + 1) \;=\; 2k + 2k \tag{1}$$

and so the average degree in G' is at least $\bar{d}/4$ (since the number of edges in G' is at least $|E|$). Recall that in such a case (i.e., when the average degree and the maximal degree are of the same order of magnitude), the definition of relative distance in the general graph model fits the definition of relative distance in the bounded-degree graph model (up to a constant factor). We stress that the bipartite gadget graphs (i.e., the G_v's) are not necessarily simple (i.e., they may have parallel edges),[10] and consequently the graph G' is not necessarily simple.

Having described the transformation of G into G', we need to address two key questions: (1) does this transformation preserve the distance from `Bipartiteness`, and (2) is this transformation local. In other words, we need to address the conditions of the definition of a local reduction, as defined in the lecture on lower bound techniques and adapted in the previous lecture.

Starting with the first question, we first note that if G is bipartite, then so is the reduced graph G'. The definition of a bipartite $\bar{d}$-regular expander, presented next, will guarantee that if G is ϵ-far from bipartite, then G' is $\Omega(\epsilon)$-far from bipartite. To motivate it suppose that G is far from being bipartite, and consider an arbitrary 2-partition of G'. In such a case, if the vertices of each bipartite expander are 2-partitioned in the natural way (i.e., all external vertices of each expander

[9]Denoting the bipartite graph by $G_v = ((X_v, I_v), E_v)$ such that $|X_v| = |I_v| = t_v$ and $E_v \subseteq X_v \times I_v$, we connect the neighbors of v to vertices in X_v, where "X" stands for external and "I" for internal. Indeed, the vertices in I_v are only connected to vertices in X_v.

[10]This will definitely happen when $\lceil d_G(v)/\bar{d}\rceil < \bar{d}$.

are assigned the same side), then the 2-partition of G' induces a 2-partition of G, which imply that the fraction of edges in G' with both endpoints on the same side is large (since the edges that connect different bipartite expanders constitute a constant fraction of the edges in G').[11] We stress that, in this case, the number of violating edges with respect to a natural 2-partition of G' equals the number of violating edges in the corresponding 2-partition of G, where an edge is called violating with respect to a 2-partition if its two endpoints are assigned to the same side.

The problem is that a general 2-partition of G' may split some of the external vertices of some bipartite expanders, and in this case it does not yield a 2-partition of G. The expansion feature defined below penalizes such a 2-partition in a way that makes it violate at least a constant fraction of the number of edges that are violated by the best natural 2-partition. This happens because such a 2-partition has many violating edges that are internal to the bipartite expanders. In particular, placing t' out of the $t \geq 2t'$ external vertices of a bipartite expander on the "wrong" side of a 2-partition may allow to avoid $t'\bar{d}$ violating edges that connect this expander to other expanders, but it causes $\Omega(t'\bar{d})$ violations inside the expander. This feature is exactly what the following definition provides.

Definition 4.1 (bipartite expanders): *We say that a family of regular bipartite graphs is c-*edge expanding *if, for every $d, t \in \mathbb{N}$, the family contains a* (not necessarily simple) *d-regular bipartite graph $((X,Y),E')$ with t vertices on each side such that for every $S \subset X$ of size at most $|X|/2$ it holds that*

$$\sum_{y \in Y} \min(|\Gamma(y) \cap S|, |\Gamma(y) \setminus S|) \geq c \cdot d \cdot |S|$$

where $\Gamma(y) \subseteq X$ denotes the multiset of neighbors of y.

The definition mandates that typical vertices in Y have many neighbors in both S and $X \setminus S$, where "many" is related to the density of S in X; that is, Definition 4.1 asserts that the average "edge mixture" (the ratio $|\Gamma(y) \cap S|/|\Gamma(y)|$ for a random $y \in Y$) is related to $|S|/|X|$. This is a slightly non-standard definition of expansion in that it refers to bipartite graphs and to edge expansion, but it follows from the standard definition of expander graphs.[12] Using c-edge expanding $\bar{d}$-regular bipartite graphs in the transformation of G into G', we observe that if G is ϵ-far from bipartite, then G' is $(c \cdot \epsilon/8)$-far from bipartite (see Exercise 3).

We now turn to the question of the locality of the transformation of G into G'. Intuitively, the transformation, which is based on local gadgets, seems very local. However, as in the previous lecture, the transformation is not local enough to fit the definition presented in the lecture on lower bound techniques. Specifically, the set of vertices of G' is not of the form $[n']$ (and is not easily put in correspondence with such a set). Nevertheless, as in the previous lecture, the transformation does fit a relaxed notion of a local reduction. Specifically, as will be shown next, it is feasible to answer incidence queries regarding G' by making few incidence queries to G, and we can afford to

[11]Recall that the number of edges that connect different bipartite expanders is $|E| = \bar{d}k/2$, whereas the number of edges in G' is at most $2\bar{d} \cdot 4k/2$.

[12]The standard definition refers to families of non-bipartite (regular) graphs of constant degree and to vertex expansion. It asserts that, for some constants $d \in \mathbb{N}$ and $c > 1$, and for each $n \in \mathbb{N}$, the family contain a d-regular n-vertex graph G_n such that any set S of at most $n/2$ vertices in G_n has at least $c \cdot |S|$ neighbors. A bipartite graph B_t as in Definition 4.1 is obtained by considering a "double cover" of G_t (i.e., replacing each vertex v in G_t by two vertices, denoted v' and v'', and connecting v' and u'' if and only if $\{u,v\}$ is an edge in G_t). Note that since d is a constant, the edge expansion of B_t follows from its vertex expansion feature (see Exercise 2). To obtain a bipartite graph of non-constant degree D, one can just duplicate each edge for D/d times.

uniformly select at random a few vertices in G'. This will suffice for emulating the execution of the tester for `Bipartiteness`, presented in the previous lecture, on the graph G'. Details follow.

First note that vertices in G' can be represented by tuples of the form $\langle v, i, \sigma\rangle$ such that $v \in [k]$, $i \in [t_v]$ and $\sigma \in \{0,1\}$ (where $\sigma = 1$ corresponds to an external vertex associated with v and $\sigma = 0$ corresponds to an internal vertex). Note, however, that determining $t_v = \lceil d_G(v)/\bar{d}\rceil$ requires determining $\bar{d}$; actually, using a reasonable approximation of $\bar{d}$ rather than its exact value does suffice for the reduction we describe here. Now, assuming we know $\bar{d}$ (or an approximation of it), given $\langle v, i, \sigma\rangle$ and $j \in \mathbb{N}$, we can tell whether $\langle v, i, \sigma\rangle$ is a vertex of G' and who is its j^{th} neighbor (by determining $d_G(v)$ and possibly determining the $(i-1)\cdot\bar{d}+j^{\text{th}}$ neighbor of v in G). Lastly, selecting a vertex in G' uniformly at random reduces to selecting at random a vertex $v \in [k]$ with probability that is proportional to t_v, which can be approximated by selecting $v \in [k]$ with probability that is proportional to $d_G(v)$. The latter task is equivalent to uniformly selecting an edge of E (see Exercise 4).

Hence, emulating the execution of the tester on G' amounts to approximating $\bar{d}$ and sampling $O(1/\epsilon)$ edges of G with probability that is sufficiently close to the uniform distribution. These tasks will be addressed in the next section. (Actually, we can avoid the task of approximating $\bar{d}$ by just trying all powers of two, and relying on the fact that the tester has one-sided error.)[13]

3 Estimating the average degree and selecting random edges

We focus on estimating the average degree (equiv., the number of edges) in a graph, while noting that similar considerations apply to selecting an edge uniformly at random. (We shall detail the required adaptations whenever they are not straightforward (i.e., in Section 3.2).) To justify the relatively high complexity of these tasks, we first present lower bounds. Specifically, we show that these tasks require $\Omega(\sqrt{k})$ queries, and we shall indeed meet these lower bound in our algorithms.

3.1 Lower bounds

For perspective, we first consider the analogous problem for functions; that is, given oracle access to a function $f : [k] \to \{0, 1, ..., k-1\}$, we wish to obtain an approximation to $\sum_{i\in[k]} f(i)$. In this case, any constant factor approximation requires $\Omega(k)$ queries to f. To see this, consider the set of functions $\{f_i : i \in [k]\}$ such that $f_i(j) = k-1$ if $j = i$ and $f_i(j) = 0$ otherwise (i.e., $j \in [k] \setminus \{i\}$). Then, an algorithm that makes $o(k)$ queries cannot distinguish a random function f_i from the all-zero function.[14]

Estimating the average degree via degree queries. In contrast to the situation with generic functions, as will be shown in Section 3.2, when $f : [k] \to \{0, 1, ..., k-1\}$ describes the degrees of

[13]The point is that using a very bad approximation of $\bar{d}$ will *not* lead the tester to reject a bipartite graph. Hence, the desired tester is obtained by invoking the original tester with all possible approximate values of $\bar{d}$ and accepting if and only if all invocations accepted. Specifically, we may invoke the original tester $\log k$ times such that in the i^{th} invocation we use 2^i as an approximation of $\bar{d}$.

[14]In light of the discussion at the end of Section 1, one may ask what happens if we confine ourselves to functions of average value at least one. In this case, one can trivially obtain a factor $k-1$ approximation, but, for any $t < k/2$, obtaining a factor t approximation requires $\Omega(k/t)$ queries. This can be shown by considering, for every t-subset $I \subset [k]$, the function $f_I(j) = k-1$ if $j \in I$ and $f_I(j) = 1$ otherwise. Note that each f_I has average value $t + 1 - (2t/k) > t$, but an algorithm that makes $o(k/t)$ queries cannot distinguish a random function f_I from the all-one function.

a simple k-vertex graph, any constant-factor approximation for a constant greater than 2 can be obtained in time $O(\sqrt{k})$. This is obtained by an algorithm that uses only degree queries, and we next observe that such an algorithm cannot obtain a better approximation factor.

Proposition 5 (limitations on approximating the average degree with degree queries): *Suppose that, algorithm A approximating the average degree of a graph $G = ([k], E)$ by making only degree queries such that*

$$\mathbf{Pr}\left[\frac{|E|}{k} \leq A^G(k) \leq \frac{2|E|}{k}\right] \geq 2/3. \tag{2}$$

Then, A makes $\Omega(k)$ queries, even if it is guaranteed that $|E| = \Theta(k)$.

Indeed, Eq. (2) refers to an approximation factor of $\frac{2|E|/k}{|E|/k} = 2$.

Proof Sketch: We show that an algorithm that makes $o(k)$ degree queries cannot distinguish the following two distributions.

1. The uniform distribution on the set of $(k-1)$-stars; that is, k-vertex graphs consisting of $k-1$ edges that are all incident to a single vertex (i.e., the graphs $G_i = ([k], \{\{i,j\} : j \in [k] \setminus \{i\}\})$, where $i \in [k]$).
2. The uniform distribution on the set of k-vertex graphs consisting of a matching of size $(k-2)/2$ and two isolated vertices; that is, k-vertex graphs containing $k-2$ vertices of degree 1 and four vertices of degree 0.

When using only degree queries, these two distributions can be distinguished only by querying one of the vertices that have degree different from 1, whereas there is only one such vertex in the first distribution and only four such vertices in the second distribution. But, when given access to a $(k-1)$-star, algorithm A is required to output a value that is at least $(k-1)/k$, whereas when given access to a matching of size $(k-2)/2$ it is required to output a value that is at most $(k-2)/k$. ■

Estimating the average degree via incidence and adjacency queries. The complexity of approximating the average degree in a graph is also lower bounded when we consider algorithms that also use incidence and adjacency queries. But here the complexity bound is $\Omega(\sqrt{k})$ (rather than $\Omega(k)$, as in the case of Proposition 5, where only degree queries were allowed but the approximation factor was required to be 2).

Proposition 6 (on the complexity of approximating the average degree): *Any constant-factor approximation algorithm for the average degree of a graph $G = ([k], E)$ must make $\Omega(\sqrt{k})$ queries, even when allowed degree, incidence and adjacency queries, and even if it is guaranteed that $|E| = \Theta(k)$.*

Proof Sketch: For any constant $\gamma > 0$, we show that an algorithm that makes $o(\sqrt{k})$ queries cannot distinguish the following two distributions.

1. The uniform distribution on k-vertex graphs consisting of a clique of size $k' = \sqrt{\gamma k}$ and an isolated matching of size $(k - k')/2$.

2. The uniform distribution on a k-vertex graphs consisting of a single perfect matching.

These two distributions can be distinguished only by making a query that refers to a vertex that belongs to the k'-clique, since any other query is answered in the same manner by both distributions (e.g., degree queries are answered by 1).[15] But, the average degree of vertices in the first distribution is $\frac{k-k'}{k} \cdot 1 + \frac{k'}{k} \cdot (k'-1) \approx 1+\gamma$, whereas the average degree of vertices in the second distribution is 1. ■

3.2 Algorithms

We show that, for every constant $\alpha > 1$, a 2α-factor approximation of the average degree can be obtained by using $O(\sqrt{k})$ degree queries, and that a α-factor approximations can be obtained by using $\widetilde{O}(\sqrt{k})$ incidence queries. These results refer to the case that the average degree is $\Omega(1)$, and a more general statement holds: Denoting the average degree by $\bar{d}$, one can obtain an arbitrary good constant factor approximation of $\bar{d}$ in *expected* time $\widetilde{O}(\sqrt{k/\bar{d}})$, which is the best possible (see Exercise 5, which extends Proposition 6). That is, when $\bar{d} \gg 1$ we actually obtain algorithms of complexity $o(\sqrt{k})$, but when $\bar{d} \ll 1$ the complexity may be $\omega(\sqrt{k})$.

Teaching note: Actually, we shall show two algorithmic approaches that obtain the stated result. The first approach (presented in Section 3.2.1) seems more intuitive, but the second approach (presented in Section 3.2.2) seems more insightful. Nevertheless, we believe that both approaches have educational benefits.

3.2.1 Bucketing vertices according to their degree

The basic idea is to partition the vertices to "buckets" according to their approximate degree, and distinguish large buckets from small buckets, where "large" means having sizxe at least $\sqrt{k}$. The key observation is that the size of large buckets can be approximated at reasonable cost, whereas there are few edges with *both* endpoints in a small bucket. Hence, *giving up on small buckets means that there are only relatively few edges that are not counted at all.* In addition, edges with a single endpoint in a small bucket are counted once, whereas edges with no endpoint in a small bucket are counted twice. This discrepancy is the reason that when using only degree queries we obtain an approximation factor of two (or rather arbitrary close to two).

Note that we can determine to which bucket a vertex belongs (by determining its degree), and estimate the size of buckets by sampling enough vertices (where for large buckets we obtain a good approximation of their size and for small buckets we only obtain an indication that they are small). Hence, by estimating the number of edges that are incident at large buckets we get close to a 2-factor approximation to the number of edges by using $O(\sqrt{k})$ degree queries. (More generally, if we are willing to tolerate an additive error term of $\binom{s}{2}$, then we can set the threshold below which a bucket is considered small to s, and make only $O(k/s)$ degree queries.)

To get a better approximation factor, we estimate, for each large bucket, the fraction of edges that have one endpoint in this bucket and the other endpoint in a small bucket. This estimation is

[15] When referring to the second distribution, it is instructive to designate k' vertices as "belonging to the k'-clique" (although there is no clique in this case). Incidence queries involving a vertex not in the k'-clique are answered with the matched neighbor if the query refers to the first neighbor and by zero otherwise. Adjacency queries regarding a pair of vertices not in the clique are answered by 1 if and only if these vertices are matched.

possible by picking random vertices in the bucket, considering a random neighbor of each of them (by using incidence queries), and determining whether this vertex reside in a small bucket.

The resulting algorithm is described next, while referring to a size parameter, denoted s (which indicates the definition of small for buckets),[16] and to an approximation parameter, denoted $\beta > 1$. The algorithm outputs approximations to the foregoing quantities (i.e., to the sizes of buckets and to the number of edges going from each large bucket to small buckets), which yield an approximation to the average degree of the input graph. Specifically, fixing an input graph $G = ([k], E)$, and letting $d_G(v)$ denote the degree of vertex v in G, for $i = 0, 1, ..., \ell \stackrel{\rm def}{=} \log_\beta k$ (so $\ell = O((\beta-1)^{-1} \log k)$, the $i^{\rm th}$ bucket is defined as $B_i = \{v \in [k] : \beta^{i-1} \leq d_G(v) < \beta^i\}$. Note that

$$\sum_{i=0}^{\ell} |B_i| \cdot \beta^{i-1} \leq \sum_{v \in [k]} d_G(v) < \sum_{i=0}^{\ell} |B_i| \cdot \beta^i \tag{3}$$

Hence, a β-factor approximations to the sizes of all the buckets yield a β^2-factor approximation to the average degree of G. Recall, however, that we shall obtain such good approximations only for large buckets (i.e., buckets of size at least s).

Algorithm 7 (a basic algorithmic scheme for estimating the average degree): *On input a graph $G = ([k], E)$, parameters $\beta > 1$ and $s \in \mathbb{N}$, and while referring to buckets B_i's such that $B_i = \{v \in [k] : \beta^{i-1} \leq d_G(v) < \beta^i\}$, proceed as follows.*

1. Estimate the sizes of the various buckets: *Take a sample of $m = \widetilde{O}(k)/s$ vertices, determine the degree of each of them and let ρ_i denote the fraction of sampled vertices that reside in B_i. If $\rho_i < s/k$, then call B_i* small*; otherwise, estimate $|B_i|$ as $\rho_i k$.*

2. Estimate the number of edges to small buckets: *For each vertex v selected in Step 1, select uniformly at random a neighbor u of v, by selecting uniformly $j \in [d_G(v)]$ and taking the $j^{\rm th}$ neighbor of v. Let $\rho'_i \leq \rho_i$ denote the fraction of sampled vertices v that reside in B_i such that their neighbour u resides in a small bucket. That is, denoting by $v_1, ..., v_m$ the sample of vertices selected in Step 1 and by $u_1, ..., u_m$ their random neighbors as selected in the current step, we let ρ'_i denote the fraction of $j \in [m]$ such that $v_j \in B_i$ and $u_j \in \cup_{i':\rho_{i'} < s/k} B_{i'}$.*

Output $\rho_0, ..., \rho_\ell$ as well as $\rho'_0, ..., \rho'_\ell$.

Note that, with probability at least 0.9, for every i it holds that B_i is declared small only if $|B_i| < (1 + o(1)) \cdot s$, and otherwise $\rho_i k \in (1 \pm o(1)) \cdot |B_i|$. Recall that the contribution of an edge to $\sum_{v \in [k]} d_G(v)$ depends on the number of endpoints it has in large buckets (i.e., buckets deemed not small): If both endpoints are in large buckets the edge contributes two units, if a single endpoint is in a large bucket the edge contributes one unit, and otherwise (i.e., no endpoint is in a large bucket) the edge contributes nothing. Hence, in this case (i.e., with high probability), we have

$$\sum_{i:\rho_i \geq s/k} \beta^{i-1} \cdot (1 - o(1)) \cdot \rho_i k \leq \sum_{v \in [k]} d_G(v) < (1 + o(1)) \cdot (\ell s)^2 + 2 \cdot \sum_{i:\rho_i \geq s/k} \beta^i \cdot (1 + o(1)) \cdot \rho_i k, \tag{4}$$

where $2 \cdot \binom{(1+o(1))\ell s}{2} < (1 + o(1)) \cdot (\ell s)^2$ is an upper bound on the contribution to $\sum_{v \in [k]} d_G(v)$ of edges with both end-points in small buckets, whereas each other edge contribute either one or two

[16] The reader may think of $s = \sqrt{k}$, but we shall consider other settings too.

units to the sum (and so the lower bound uses one unit and the upper bound uses two). Hence, if $(\ell s)^2 = o(1) \cdot \sum_{v\in[k]} d_G(v)$, then Eq. (4) yields a $2 \cdot (\beta + o(1))$-factor approximation to the average degree of G. Note that this approximation is obtained based on Step 1 only, which only uses degree queries. But using the quantities estimated in Step 2, we can do better.

Specifically, recall that $\beta^{i-1} \cdot (1 - o(1)) \cdot \rho_i k$ was used in Eq. (4) as a lower bound on the contribution of vertices in (a large) B_i to $\sum_{v\in[k]} d_G(v)$ whereas $2 \cdot \beta^i \cdot (1 + o(1)) \cdot \rho_i k$ was used as an upper bound. But assuming that ρ'_i/ρ_i estimates the average over $v \in B_i$ of the fraction of neighbors of v that reside in small buckets, we get much tigher bounds: The contribution of vertices in a large B_i to $\sum_{v\in[k]} d_G(v)$ is at least $(1-o(1))\cdot(\rho_i+\rho'_i)\cdot\beta^{i-1}k$ and at most $(1+o(1))\cdot(\rho_i+\rho'_i)\cdot\beta^i k$, since ρ'_i represents the "lost" contribution of edges with one endpoint in B_i and one endpoint in a small bucket. This argument is captured in the following claim and further detailed in its proof.

Claim 8 (the core of the analysis of Algorithm 7): *Suppose that for every i it holds that if $\rho_i < s/k$, then $|B_i| < (1 + o(1)) \cdot s$, and otherwise $\rho_i k \in (1 \pm o(1)) \cdot |B_i|$. Further suppose that if $\rho_i \geq s/k$, then*

$$\rho'_i = (1 \pm o(1)) \cdot \frac{1}{k} \cdot \sum_{v\in B_i} \frac{d'_G(v)}{d_G(v)},$$

where $d'_G(v)$ denotes the number of neighbors of v in $S = \cup_{i':\rho_{i'}<s/k} B_{i'}$. Then,

$$(1-o(1))\cdot \sum_{i:\rho_i\geq s/k} (\rho_i+\rho'_i)\cdot\beta^{i-1}k \;<\; \sum_{v\in[k]} d_G(v) \;<\; (1+o(1))\cdot(\ell s)^2+(1+o(1))\cdot \sum_{i:\rho_i\geq s/k} (\rho_i+\rho'_i)\cdot\beta^i k.$$

Proof: The claim reduces to proving

$$\sum_{v\in[k]} d_G(v) = 2 \cdot |E(S,S)| + \sum_{i:\rho_i\geq s/k} \sum_{v\in B_i} (d_G(v) + d'_G(v)) \tag{5}$$

where $E(S,S)$ denotes the set of edges with both endpoints in S. To prove Eq. (5), we consider the contribution of each edge $\{v,u\}$ to each of its sides. First note that each edge contributes exactly two units to the l.h.s of Eq. (5). Now, we consider its contribution to the r.h.s of Eq. (5), by distinguishing three cases.

1. If both endpoints of $\{u,v\}$ are in S, then $\{u,v\}$ contributes two units to $2 \cdot |E(S,S)|$ and nothing to the sum.

2. If exactly one endpoint of $\{u,v\}$ is in S, then $\{u,v\}$ contributes nothing to $2 \cdot |E(S,S)|$ and contributes two units to the sum, since it contributes one unit to $d_G(v)$ and one unit to $d'_G(v)$, where we assume (w.l.o.g.) that $u \in S$.

3. If no endpoint of $\{u,v\}$ is in S, then $\{u,v\}$ contributes nothing to $2\cdot|E(S,S)|$ and contributes two units to the sum, since it contributes one unit to $d_G(v)$ and one unit to $d_G(u)$, while contributing nothing to $d'_G(v) + d'_G(u)$.

Hence, in all cases, the edge $\{u,v\}$ contributes two units to the r.h.s of Eq. (5). In fact, the argument can be summarized by writing

$$\sum_{v\in[k]} d_G(v) = 2 \cdot |E(S,S)| + \sum_{v\in[k]\setminus S} d_G(v) + \sum_{v\in[k]\setminus S} d'_G(v), \tag{6}$$

but Eq. (5) is more instructive towards finishing the proof. Indeed, the claim follows since $2 \cdot |E(S,S)| < |S|^2$, whereas for $i \in [k]$ such that $\rho_i \geq s/k$ it holds that $\sum_{v \in B_i} d_G(v) \approx |B_i| \cdot \beta^i \approx \rho_i k \cdot \beta^i$ and $\sum_{v \in B_i} d'_G(v) \approx \rho'_i k \cdot \beta^i$. ■

Digest. The idea underlying Algorithm 7 is that all vertices in the same bucket have approximately the same contribution to $\sum_{v \in [k]} d_G(v)$ (i.e., each vertex in B_i contributes approxmately β^i). Hence, approximating all $|B_i|$'s yields an approxmation to the said sum. The problem is that we cannot afford to approxuimate all $|B_i|$'s well enough, since some B_i may be too small. Fortunately, as shown in the foregoing discussion, a good approximation of the sizes of the large B_i's (which we can afford) suffices for a factor two approximation of the sum, since there are very few edges that have both endpoints in small buckets. Getting a better approximation requires approximating the fraction of edges that have a single endpoint in a large bucket, and this can actually be done without attributing these edges to the specific large buckets (although Algorithm 7 did use such an attribution). The last assertion hints that the bucketing is actually not so important; what is important is the handling the case that a small set of vertices have many incident edges. This observation will become more explicit in Section 3.2.2.

Teaching note: The rest of Section 3.2.1 is a bit tedious, and can be skipped if under time pressure. In such a case, we recommend leaving it for independent reading, since it does make two important points. The first point is that we can set the threshold s "adaptively" rather than rely on its being given to us from the outside. The technique used here is quite generic and good to know (see also Exercises 6 and 7). The second point is using the output provided by Algorithm 7 in order to sample random edges in the graph.

Setting the parameters. As apparent in Claim 8, the parameter $\beta > 1$ determines the quality of the approximation, and it can be set to a constant that is arbitrarily close to 1. This means that $\ell = \log_\beta k = O(\epsilon^{-1} \log k)$, where $\epsilon \stackrel{\text{def}}{=} \beta - 1 > 0$ is a positive constant. Hence, for constant $\beta > 1$, we have $\ell = O(\log k)$.

The setting of the parameter s is less obvious. One the one hand, we should set s such that $(\ell s)^2 < \epsilon \cdot |E|$, which guarantees that the omission of edges with both endpoints in small buckets has little effect on the quality of the approximation of $|E|$. On the other hand, we should set s as large as possible (subject to $(\ell s)^2 < \epsilon \cdot |E|$), since the complexity of Algorithm 7 is inversely proportional to s. Specifically, recall that the time complexity of Algorithm 7 is $\widetilde{O}(k)/s$, so under an optimal setting (i.e., $s = \Omega(\sqrt{\epsilon \cdot |E|}/\ell)$) we get a complexity bound of $\widetilde{O}(k)/\sqrt{\epsilon \cdot |E|}$, which equals $\widetilde{O}(\sqrt{k})/\sqrt{\bar{d}}$, since $\epsilon > 0$ is a constant and $\bar{d} = 2|E|/k$.

If $|E| = \Omega(k)$, then we can use $s = \Omega(\sqrt{k}/\log k)$ and get a complexity bound of $\widetilde{O}(\sqrt{k})$. However, if we have a higher lower bound on $|E|$, then we get a better complexity bound. In general, if we have a rough estimate of $|E|$, then we can set s accordingly (i.e., $s = \Theta(\sqrt{L}/\log k)$, where L is the lower bound provided to us) and obtain a good estimate of $|E|$.

Actually, we can avoid the use of an *a priori* lower bound on $|E|$, and obtain an algorithm with a complexity bound that depends on the actual value of $|E|$. This is done by iteratively invoking Algorithm 7 using guesses for $|E|$ that are cut by half in each iteration, while relying on the fact that (with high probability) Algorithm 7 does not overestimate the value of $|E|$ (regardless of the value of s that is used).[17] Let us spell out the result obtained.

Theorem 9 (approximating the average degree with incidence queries): *For every constant $\alpha > 1$, there exists an algorithm that approximates the average degree, $\bar{d}$, in a given k-vertex graph to within a factor of α in expected time $\widetilde{O}(\sqrt{k/\bar{d}})$.*

Note that the time bound provided in Theorem 9 refers to the expectation, while admitting that much longer executions are possible (typically when some approximations fail, which happens with small probability).

Proof Sketch: We shall only prove a weaker bound of $\widetilde{O}(\sqrt{k})/\sqrt{\bar{d}}$, while noting that obtaining the better bound is possible by a small modification of Algorithm 7.[18] On input $G = ([k], E)$, we proceed in iterations such that in the i^{th} iteration, guessing that $\bar{d} \approx k/2^{i-1}$, we invoke Algorithm 7 while using $s = \sqrt{k^2/2^i}/O(\ell) = k/O(2^{i/2}\ell)$. (We also apply error reduction so that the probability the algorithm provides an overestimate of $|E|$ in any iteration is at most 1/3.)[19] Actually, in the i^{th} iteration we use the hypothesis that $\bar{d} \geq k/2^{i-1}$, while noting that if this hypothesis is correct then (w.h.p.) we obtain a good estimate of $\bar{d}$, and in any case (w.h.p.) we do not get an over estimate of $\bar{d}$. If the i^{th} iteration outputs an estimate (for $\bar{d}$) that is larger than $k/2^{i-1}$, then we output it and halt, since (w.h.p.) the algorithm never overestimates, which implies that our current hypothesis $\bar{d} \geq k/2^{i-1}$ was correct and so (w.h.p) the output is a good estimate of $\bar{d}$. Otherwise, we proceed to the next iteration.

Hence, with high constant probability, we halt by iteration $i = \log(2k/\bar{d})$, which has complexity $\widetilde{O}(k)/(k/2^{i/2}) = \widetilde{O}(\sqrt{k})/\sqrt{\bar{d}}$. The claim about the expected running time follows by observing that invoking Algorithm 7 with smaller than required value of s (i.e., the value $s = \Theta(\sqrt{k\bar{d}})/\ell$) yields error probability that is exponentially decreasing in $\sqrt{k\bar{d}}/s$. Alternatively, see Exercise 7. ■

Sampling. In order to sample uniformly at random an edge in the graph $G = ([k], E)$, we first approximate the number of edges as described in the proof of Theorem 9. Recall that this approximation procedure also provides us with the identity of the large buckets and their approximate sizes, denoted by a_i's. Note that within the very same complexity bound, we can sample vertices uniformly at random from each of the large buckets. Hence, we can sample vertices in the large buckets at random according to their degree, by first selecting a bucket B_i with probability proportional to $a_i \cdot \beta^i$, and then select a vertex $v \in B_i$. Lastly, with probability one half we output v, and with probability one half we select uniformly a neighbor of v, denoted u, and output u if and only if u resides in a small bucket (i.e., we output nothing if the neighbor u resides in a large bucket). This description is to be understood within the repeated sampling paradigm, where in case no output is generated the procedure is repeated. Letting L denote the set of large buckets

[17] That is, the bound $(1 - o(1)) \cdot \sum_{i:\rho_i \geq s/k}(\rho_i + \rho'_i) \cdot \beta^{i-1}k < \sum_{v\in[k]} d_G(v)$ holds regardless of the value of the parameter s; see Claim 8.

[18] **Advanced comment:** Basically, the source of trouble is that the analysis of the algorithm referred to $\ell = \log_\beta k$ buckets and to the setting of $s = \sqrt{k\bar{d}}/\ell$, whereas it suffices to consider the $\ell' \stackrel{\text{def}}{=} \log_\beta(2k/\bar{d})$ buckets that contain vertices of degree at least $\bar{d}/2$ and to use $s = \sqrt{k\bar{d}}/\ell'$. Alternatively, one can prove the better bound by using the approach presented in Section 3.2.2.

[19] Since there are only $2\log k$ iterations, the cost of such an error reduction is rather small. Furthermore, we can apply non-identical levels of error reduction in the various iterations so that in the i^{th} iteration the error probability is at most $1/(i+3)^2$. Doing so allows to have a smaller overhead in the first iterations, which have lower complexity.

and $M = \sum_{i \in L} a_i \cdot \beta^i$, observe that (in each iteration of this sampling procedure) each vertex v that resides in a large bucket B_i is output with probability

$$\begin{aligned}
\mathbf{Pr}[i \text{ chosen}] \cdot \mathbf{Pr}[v \text{ chosen in } B_i] \cdot \mathbf{Pr}[v \text{ is output}] &= \frac{a_i \cdot \beta^i}{M} \cdot \frac{1}{|B_i|} \cdot \frac{1}{2} \\
&\approx \frac{d_G(v)}{M} \cdot \frac{1}{2} \\
&= \frac{|E|}{M} \cdot \frac{d_G(v)}{2|E|}
\end{aligned}$$

where the approximation is due to $a_i \approx |B_i|$ and $\beta^i \approx d_G(v)$. Similarly, the probability that such an iteration outputs a vertex u that resides in a small bucket equals

$$\begin{aligned}
&\mathbf{Pr}[\text{a neighbor of } u \text{ is chosen in a large bucket}] \cdot \mathbf{Pr}[u \text{ is output}] \\
&\quad= \sum_{i \in L} \sum_{v \in \Gamma(u) \cap B_i} \mathbf{Pr}[v \text{ is chosen}] \cdot \mathbf{Pr}[u \text{ is output}] \\
&\quad= \sum_{i \in L} \sum_{v \in \Gamma(u) \cap B_i} \left(\frac{a_i \cdot \beta^i}{M} \cdot \frac{1}{|B_i|}\right) \cdot \left(\frac{1}{2} \cdot \frac{1}{d_G(v)}\right) \\
&\quad\approx \sum_{i \in L} \sum_{v \in \Gamma(u) \cap B_i} \frac{1}{2M} \\
&\quad\approx \frac{|E|}{M} \cdot \frac{d''_G(u)}{2|E|}
\end{aligned}$$

where $\Gamma(u)$ denotes the set of u's neighbors and $d''_G(u) = |\Gamma(u) \setminus S| \leq d_G(u)$ denotes the number of neighbors of u that reside in large buckets (while recalling that S denotes the set of vertices that reside in small buckets). Recall that $\sum_{u \in S}(d_G(u) - d''_G(u)) = 2 \cdot |E(S,S)| \ll |E|$, which implies that typically $d''_G(u) \approx d_G(u)$. Hence, a single iteration produces an output with probability approximately $|E|/M \geq 0.5 - o(1)$, and the output distribution is close to the distribution of vertices selected in proportion to their degree.

3.2.2 Sorting vertices according to their degree

The bottom-line of the digest provided in the middle of Section 3.2.1 is that, when approximating the average degree of a graph, the main issue is handling the case that a small set of vertices has many incident edges. The observation that fuels the current approach is that these vertices are necessarily those of the highest degree. Hence, rather that setting aside buckets of size at most s, we set aside the s vertices that have highest degree (breaking ties arbitrarily).

To see what is gained by setting aside these high degree vertices, let us consider a naive approach to approximating $\bar{d}$ (i.e., $\sum_{v \in [k]} d_G(v)/k$). This approach consists of selecting m random vertices and using their average degree as an estimator to $\bar{d}$. Now, let the random variable ζ_i denote the result of the i^{th} experiment; that is, $\zeta_i = d_G(v)$, where v is uniformly distributed in $[k]$. Then, $\mathbb{E}[\zeta_i] = \bar{d}$ obviously holds, but the problem is that $\mathbb{V}[\zeta_i]$ can only be upper-bounded by $k \cdot \bar{d}$ (whereas in some cases a lower bound of $\Omega(k \cdot \bar{d})$ does hold).[20] The point is that, when using a law of large

[20]Here (and in the rest of this exposition), we use the fact that, for any random variable $Z \in [0, B]$, it holds that

numbers (e.g., Chernoff Bound), we need to set $m = \Omega\left(\frac{\mathbb{V}[\zeta_i]}{\mathbb{E}[\zeta_i]^2}\right) = \Omega(k/\bar{d})$. In particular, if $\bar{d} = \Theta(1)$, then we get $m = \Omega(k)$, which is useless.

Denoting the set of s vertices of highest degree by H, let us now see what happens when we approximate $\bar{d}$ by $\sum_{v\in[k]\setminus H} d_G(v)/k$, where the latter term is approximated by sampling.[21] That is, suppose that we select m random vertices and use as our estimate the average contribution to the foregoing sum. Let ζ'_i denote the result of the i^{th} experiment; that is, $\zeta'_i = d_G(v)$ if $v \in [k]\setminus H$ and $\zeta'_i = 0$ otherwise, where v is uniformly distributed in $[k]$. Note that $\mathbb{E}[\zeta'_i] \le \bar{d}$ and $\mathbb{E}[\zeta'_i] \ge (|E| - s^2)/k = 0.5\bar{d} - (s^2/k) \approx 0.5\bar{d}$, provided that $s = \sqrt{k\bar{d}}/O(1)$. The good news are that $\mathbb{V}[\zeta'_i]$ can be upper-bounded by $\max_{v\in[k]\setminus H}\{d_G(v)\}\cdot\bar{d} \le k\cdot\bar{d}^2/s$, since $\max_{v\in[k]\setminus H}\{d_G(v)\} \le \min_{v\in H}\{d_G(v)\} \le k\cdot\bar{d}/s$ (which holds because $\sum_{v\in H} d_G(v) \le k\cdot\bar{d}$ and $|H| = s$). Hence, when using a law of large numbers, we can set $m = O\left(\frac{\mathbb{V}[\zeta_i]}{\mathbb{E}[\zeta_i]^2}\right) = O\left(\frac{k\cdot\bar{d}^2/s}{\bar{d}^2}\right) = O(k/s)$, provided that $\mathbb{E}[\zeta'_i] > \bar{d}/3$, which holds when $s^2 < \bar{d}k/3$. In particular, if we pick $s = \sqrt{k\bar{d}}/O(1)$, then we get a constant (larger than two) factor approximation using $m = O(\sqrt{k/\bar{d}})$.

Note that the foregoing procedure assumes that we can tell whether or not a sampled vertex is in H (i.e., is among the s vertices of highest degree). While it is not clear how to determine the exact ranking of vertices according to this order, we can approximate their rank based on the degrees of the sampled vertices, and such approximation will suffice. However, this issue will disappear in the modification presented next, which is aimed at reducing the approximation factor from (a constant arbitrary close to) two to a constant arbitrary close to one. Recall that the source of problem is that we only have the following bounds

$$|E| - 2\cdot|E(H,H)| \;\le\; \sum_{v\in[k]\setminus H} d_G(v) \;\le\; 2\cdot|E|, \tag{7}$$

where the additive loss of $2\cdot|E(H,H)|$ is due to edges with both endpoints in H and the factor of two loss is due to edges with one endpoint in H that are counted (only) at the endpoint that resides in $[k]\setminus H$. The new idea is to count edges only at the endpoint that has lower degree (while breaking ties arbitrarily). Specifically, let $\vec{d}_G(v)$ denote the number of neighbours of v that have rank higher than v (i.e., $\vec{d}_G(v) = |\{u \in \Gamma(v) : (d_G(u), u) > (d_G(v), v)\}|$).[22] Then, $\sum_{v\in[k]} \vec{d}_G(v) = |E|$ and

$$|E| - |E(H,H)| \;\le\; \sum_{v\in[k]\setminus H} \vec{d}_G(v) \;\le\; |E|. \tag{8}$$

(This observation holds for any way of assigning edges to one of their endpoints.) The key observation is that $\vec{d}_G(v) \le \sqrt{2|E|}$ holds for every $v \in [k]$. In particular, if G is sparse, then $\vec{d}_G(v) \ll k$ for every v.

Claim 10 (bounding $\vec{d}_G(v)$): *For every vertex v in the graph $G = ([k], E)$, it holds that $\vec{d}_G(v) \le \sqrt{2|E|}$.*

$\mathbb{V}[Z] \le \mathbb{E}[Z^2] \le B\cdot\mathbb{E}[Z]$. In general, this inequality is tight (e.g., when $\mathbf{Pr}[Z = B] = p \le 1/2$ and $\mathbf{Pr}[Z = 0] = 1 - p$, we get $\mathbb{E}[Z] = pB$ and $\mathbb{V}[Z] = p\cdot B^2 - (p\cdot B)^2 \ge B\cdot\mathbb{E}[Z]/2$). This inequality is tight also when Z represents the degree distribution in a graph. For example, generalizing the proof of Proposition 5, consider the graph is $K_{t,k-t}$. Then, letting ζ_i be as above, we have $\bar{d} < 2t$ and $\mathbb{V}[\zeta_i] > \frac{t}{k}\cdot(k - t - \bar{d})^2 > t\cdot(k - 3t)^2/k$, which is $\Omega(tk)$ when $t < k/4$.

[21] Obtaining such an approximation is not obvious (and will be discussed later).

[22] Note that the definition of $\vec{d}_G(v)$ breaks ties (in the ranking according to degrees) by using a lexicographic order on pairs consisting of the vertex's degree and its label.

Proof: If v is one of the first $\sqrt{2|E|}$ vertices according to the foregoing order, then the claim holds since $\vec{d}_G(v)$ only counts edges that go to the higher ranked vertices. But otherwise (i.e., at least $\sqrt{2|E|}$ vertices have higher ranking than v), it must holds that $\vec{d}_G(v) \leq d_G(v) \leq \sqrt{2|E|}$, since (by the hypothesis) the number of higher ranked vertices is at least $\sqrt{2|E|}$, whereas the degree of each of them is at least $d_G(v)$, which implies $\sqrt{2|E|} \cdot d_G(v) \leq 2|E|$. ■

In light of the foregoing, for any $\epsilon > 0$, we can obtain a factor $(1+\epsilon)$ approximation of $|E|$ by selecting a sample of $m = O(\sqrt{2|E|}/\epsilon^2\bar{d})$ vertices, denoted S, and using $Z \stackrel{\text{def}}{=} \sum_{v\in S} \vec{d}_G(v)/m$ as our estimate. Letting ζ_i'' denote the result of the ith experiment (i.e., $\zeta_i'' = \vec{d}_G(v)$ for a uniformly distributed $v \in [k]$), we have $\mathbb{E}[Z] = \mathbb{E}[\zeta_i''] = \frac{1}{k} \cdot \sum_{v\in[k]} \vec{d}_G(v) = \bar{d}/2$ and $\mathbb{V}[Z] = \mathbb{V}[\zeta_i'']/m \leq \sqrt{2|E|} \cdot \mathbb{E}[\zeta_i'']/m$ (where we use $\mathbb{V}[\zeta_i''] \leq \mathbb{E}[(\zeta_i'')^2] \leq \max_{v\in[k]}\{\vec{d}_G(v)\} \cdot \mathbb{E}[\zeta_i'']$). Hence,

$$\begin{aligned}
\mathbf{Pr}[|Z - 0.5\bar{d}| \geq 0.5\epsilon \cdot \bar{d}] &\leq \frac{\mathbb{V}[Z]}{(0.5\epsilon\bar{d})^2} & (9)\\
&\leq \frac{\sqrt{2|E|} \cdot 0.5\bar{d}/m}{(0.5\epsilon\bar{d})^2}\\
&= \frac{\sqrt{8|E|}}{\epsilon^2\bar{d} \cdot m} & (10)
\end{aligned}$$

which can be made an arbitrary small positive constant by setting $m = O(\sqrt{8|E|}/\epsilon^2\bar{d})$ appropriately. The "only" problem is that it is not clear how to compute $\vec{d}_G$. Nevertheless, we can approximate $\sum_{v\in S} \vec{d}_G(v)$ by computing $\sum_{v\in S} d_G(v)$ and approximating $\rho \stackrel{\text{def}}{=} \frac{\sum_{v\in S} \vec{d}_G(v)}{\sum_{v\in S} d_G(v)}$, since $\sum_{v\in S} \vec{d}_G(v) = \rho \cdot \sum_{v\in S} d_G(v)$. Specifically, we approximate ρ by sampling the edges that are incident at S (i.e., pairs (v,u) such that $v \in S$ and $\{v,u\} \in E$), and computing the the fraction of pairs such that $(d_G(v), v) < (d_G(u), u)$. This works, provided that the foregoing fraction (i.e., $\rho = \frac{\sum_{v\in S} \vec{d}_G(v)}{\sum_{v\in S} d_G(v)}$) is not too small, which can be guaranteed by lower-bounding the numerator and upper-bounding the denumerator. As noted in Eq. (9)&(10), the numerator is "well concentrated" around its mean, but dealing with expression in the denominator is what we were trying to avoid. Still a very weak bound, of the type that is provided by Markov's inequality, will suffice here (i.e., the value of $\sum_{v\in S} d_G(v)$ cannot be *much larger* than its expectation). Details follow.

Starting with the actual algorithm, we let m' denote a generic parameter; the reader may think of the case that m' equals $\Theta(\sqrt{|E|}/\bar{d}) = \Theta(\sqrt{k/\bar{d}})$.

Algorithm 11 (an alternative algorithmic scheme for estimating the average degree): *On input a graph $G = ([k], E)$ and parameters m' and ϵ, proceed as follows.*

1. Take a primary sample (of uniformly distributed vertices): *Select uniformly at random $m = O(m'/\epsilon^2)$ vertices, and let $\{v_1, ..., v_m\}$ denote the resulting multiset. Using degree queries, compute $D \leftarrow \sum_{i\in[m]} d_G(v)$.*

2. Take a secondary sample of edges: *For $j = 1, ..., t \stackrel{\text{def}}{=} O(1/\epsilon^3)$, select $i_j \in [m]$ such that* $\mathbf{Pr}[i_j = i] = d_G(v_i)/D$*, and select u_{i_j} uniformly at random among the neighbours of v_{i_j}. Let J denote the set of $j \in [t]$ such that $(d_G(v_{i_j}), v_{i_j}) < (d_G(u_{i_j}), u_{i_j})$.*

The value of $\frac{|J|}{t} \cdot \frac{D}{m}$ can be output as an estimate of $|E|/k$, and a uniformly chosen edge in $\{\{v_{i_j}, u_{i_j}\} : j \in J\}$ can be output as a sampled edge of the graph.

We first note that if $m' \geq c \cdot \sqrt{8|E|/\bar{d}}$, then $\frac{1}{m} \cdot \sum_{i\in[m]} \vec{d}_G(v_i) = (1 \pm \epsilon) \cdot \frac{|E|}{k}$ with probability at least $1-(1/c)$. (This was essentially shown in the motivating discussion that preceded Algorithm 11 (see Eq. (9)&(10)), where Z represented the average of the $\vec{d}_G(v_i)$'s and $0.5\bar{d}$ was used instead of $|E|/k$.) We next observe that the expected value of D is $m \cdot 2|E|/k$, and hence $\mathbf{Pr}[D > 2m|E|\epsilon^{-1}/k] < \epsilon$. Assuming that $D \leq 2m|E|\epsilon^{-1}/k$, note that

$$\frac{\sum_{i\in[m]} \vec{d}_G(v_i)}{D} > \frac{(1-\epsilon)\cdot(m|E|)/k}{2m|E|\epsilon^{-1}/k} = \frac{1-\epsilon}{2\epsilon^{-1}} = \Omega(\epsilon).$$

Hence, sampling $O(\epsilon^{-3})$ pairs uniformly in $\{(v_i,u) : i\in[m] \wedge \{u,v_i\}\in E\}$ yields an $(1+\epsilon)$-factor approximation of $\sum_{i\in[m]} \vec{d}_G(v_i)$. Specifically, with high probability,

$$\frac{|J|}{t} = (1 \pm \epsilon) \cdot \frac{\sum_{i\in[m]} \vec{d}_G(v_i)}{D}.$$

Recalling that $\frac{1}{m} \cdot \sum_{i\in[m]} \vec{d}_G(v_i) = (1 \pm \epsilon) \cdot \frac{|E|}{k}$, we conclude that

$$\frac{|J|}{t} \cdot \frac{D}{m} = (1 \pm \epsilon) \cdot \frac{\sum_{i\in[m]} \vec{d}_G(v_i)}{m} = (1 \pm \epsilon)^2 \cdot \frac{|E|}{k}.$$

Lastly, we turn to the analysis of the sampling feature provided by Algorithm 11 (when $m' \geq c \cdot \sqrt{8|E|/\bar{d}}$ for a sufficiently large constant c). Letting $\vec{\Gamma}_G(v)$ denote the set of neighbors of v with rank higher than v (i.e., those counted in $\vec{d}_G(v)$), we first observe that each edge $\{u,v\}$ appears in $E' \stackrel{\rm def}{=} \{\{v_i,u'\}\in E : i\in[m] \;\wedge\; u' \in \vec{\Gamma}_G(v_i)\}$ with probability $1-(1-1/k)^m \approx m/k$, since the edge appears in E' if and only if its lower ranked endpoint is selected in the primary sample $\{v_1,...,v_m\}$. Next note that, conditioned on v appearing in the primary sample and on $|E'|/m = (1\pm\epsilon)\cdot|E|/k$, the edge $\{u,v\} \in E'$ is selected (for output) with probability $1/|E'|$. Recall that, with high probability, it holds that $|E'|/m = (1 \pm \epsilon) \cdot |E|/k$, since $\frac{1}{m} \cdot \sum_{i\in[m]} \vec{d}_G(v_i) = (1 \pm \epsilon) \cdot \frac{|E|}{k}$. Hence, each edge appears as output with probability approximately $(m/k) \cdot |E'|^{-1} = (1 \pm \epsilon)/|E|$.

4 Using adjacency queries: the case of Bipartiteness

Two natural questions arise regarding the tester for `Bipatiteness` asserted in Theorem 4: Firstly, recall that in case the input graph is dense (i.e., $|E| = \Omega(k^2)$), testing `Bipartiteness` is possible within complexity that is independent of the size of the graph, whereas this is not reflected in Theorem 4. In other words, in light of the results regarding the dense graph model (let alone their contrast with the results for the bounded-degree graph model), one may suspect that the complexity of testing `Bipartitness` in the general graph model may be related to the density of edges in the input graph, whereas Theorem 4 does not relate to the edge density. Secondly, we note that the algorithm used in the proof of Theorem 4 only uses incidence queries, whereas the model allows also adjacency queries.

The issues raised by these two questions are actually related. As shown in [13], for every k and $\rho = \rho(k) \in (\Omega(1/k), 1)$, a tester for `Bipatiteness` (of k-vertex graphs) that *only makes incidence queries* must have query complexity $\Omega(\sqrt{k})$ even when guaranteed that the edge density in the input graph is $\Theta(\rho)$. On the other hand, we observe that using adjacency queries (only), allows to emulate the tester for the dense graph model within complexity that only depends on the edge density. This is actually a generic result.

Theorem 12 (emulating testers for the dense graph model (in the general graph model)): *Let T be a tester of the graph property* Π in the dense graph model*, and let $q : \mathbb{N} \times [0,1] \to \mathbb{N}$ denote its query complexity. Then,* Π *can be tested* in the general graph model *such that the expected query complexity of ϵ-testing the input graph $G = ([k], E)$ is $q(k, 0.9\rho \cdot \epsilon) + \widetilde{O}(1/\rho)$, where $\rho \stackrel{\rm def}{=} 2|E|/k^2$. Furthermore, the resulting tester preserves one-sided error and only uses adjacency queries.*

The fact that the complexity bound refers to the expectation is due to the need to approximate ρ (and this is also the source of the $\widetilde{O}(1/\rho)$ term). If we know a lower bound ρ' on ρ, then we can ϵ-test the input graph $G = ([k], E)$ using exactly $q(k, \rho' \cdot \epsilon)$ queries.

Proof Sketch: We first observe that using $\widetilde{O}(1/\rho)$ random adjacency queries, we can approximate ρ up to any desired constant factor (where the point is actually getting a good lower bound ρ' on ρ). This is done in iterations such that in the $i^{\rm th}$ iteration we try to confirm that $\rho \approx 2^{-i+0.5}$. Using $O(i \cdot 2^i)$ random queries in the $i^{\rm th}$ iteration, we can upper-bound the error probability of iteration i by $0.1 \cdot 2^{-i}$.

Having obtained an approximation $\widetilde{\rho}$ to ρ, we invoke the tester T with proximity parameter $\widetilde{\rho} \cdot \epsilon$, where ϵ is the proximity parameter given to us and (w.l.o.g.) $\rho \geq \widetilde{\rho}$. The point is that a proximity-parameter value of ϵ in the general graph model, where we normalize by $|E| = \rho k^2/2$, corresponds to a proximity-parameter value of $\rho\epsilon$ in the dense graph model (where we normalize by $k^2/2$). ■

Back to the special case of `Bipartiteness`. Applying Theorem 12 to the `Bipartiteness` tester (of the dense graph model), we derive a tester of (query and time) complexity $\widetilde{O}(1/\epsilon\rho)^2$ for the general graph model.[23] But this result is not optimal: An alternative approach, to be presented next, yields a tester of (query and time) complexity $\text{poly}(\epsilon^{-1} \log k) \cdot \rho^{-1}$.

In light of the ideas presented in Section 2.2, we shall focus on *the case that the maximal degree of the input graph is of the same order of magnitude as its average degree*; that is, we assume that the maximal degree is $O(\bar{d})$, where $\bar{d}$ denotes the average degree of the input graph. Furthermore, we assume that the algorithm is given an upper bound, denoted d, on the maximal degree, and that $d = O(\bar{d})$. The following algorithm is a variant of the `Bipartite` tester presented in the previous lecture. It differs in the number of random walks that it takes from each vertex (as determined by m), and in what it does with the sets R_0 and R_1 (see Step 2c).

Algorithm 13 (an alternative algorithm for testing `Bipartiteness` (in the general graph model)): *On input d, k, ϵ and oracle access to incidence and adjacency functions of a k-vertex graph, $G = ([k], E)$, of degree bound d, repeat $t \stackrel{\rm def}{=} \Theta(\frac{1}{\epsilon})$ times:*

1. *Uniformly select s in $[k]$.*
2. (Try to find an odd-length cycle through vertex s):
   (a) *Perform $m \stackrel{\rm def}{=}$ $\text{poly}(\epsilon^{-1} \log k) \cdot \sqrt{k/d}$ random walks starting from s, each of length $\ell \stackrel{\rm def}{=}$ $\text{poly}(\epsilon^{-1} \log k)$.*[24]

[23]Recall that the best ϵ'-tester for the dense graph model has time complexity $\widetilde{O}(1/\epsilon')^2$ and that $\Omega(1/\epsilon')^{3/2}$ is a lower bound on the query complexity in this case.

[24]Recall that a random walk of length ℓ starting at s is a path $(s = v_0, v_1, ..., v_\ell)$ in G selected at random such that v_i is uniformly distributed among the neighbors of v_{i-1}.

(b) *Let* R_0 (respectively, R_1) *denote the set of vertices reached from* s *in an even* (respectively, odd) *number of steps in any of these walks. That is, assuming that* ℓ *is even, for every such walk* $(s = v_0, v_1, ..., v_\ell)$*, place* $v_0, v_2, ..., v_\ell$ *in* R_0 *and place* $v_1, v_3, ..., v_{\ell-1}$ *in* R_1*.*

(c) *For every* $\sigma \in \{0,1\}$ *and* $u, v \in R_\sigma$*, if* $\{u,v\}$ *is an edge in* G*, then* reject.

If the algorithm did not reject in any of the foregoing t *iterations, then it* accepts.

Note that Step 2a is implemented by using incidence queries, whereas Step 2c is implemented using adjacency queries. The time (and query) complexity of Algorithm 13 is $t \cdot ((m \cdot \ell)^2 + m \cdot \ell \cdot \log d) = \text{poly}(\epsilon^{-1} \log k) \cdot (k/d)$, where the $\log d$ factor is due to determining the degree of each vertex encountered in the random walk. It is evident that the algorithm always accepts a bipartite graph.

As in the previous lecture, the core the analysis is proving that *if the input graph is* ϵ*-far from being bipartite and* $\bar{d} = \Omega(d)$*, then Algorithm 13 rejects with probability at least* $2/3$.[25] Again, we confine ourselves to the "rapid mixing" case, and consider a single execution of Step 2, starting from an arbitrary vertex s, and using lazy random walks instead of the natural random walks that are used in the algorithm. (For sake of self-containment, we reproduce the relevant definitions next.)

Definition 13.1 (lazy random walks and the rapid mixing feature): *Let* $(v_1, ..., v_\ell) \leftarrow \mathcal{RW}_\ell$ *be an* ℓ-step lazy random walk (on $G = ([k], E)$) *starting at* $v_0 \stackrel{\text{def}}{=} s$*; that is, for every* $\{u,v\} \in E$ *and every* $i \in [\ell]$*, it holds that*

$$\begin{aligned} \mathbf{Pr}_{(v_1,...,v_\ell)\leftarrow\mathcal{RW}_\ell}[v_i = v | v_{i-1} = u] &= \frac{1}{2d} & (11)\\ \mathbf{Pr}_{(v_1,...,v_\ell)\leftarrow\mathcal{RW}_\ell}[v_i = u | v_{i-1} = u] &= 1 - \frac{d_G(u)}{2d} & (12) \end{aligned}$$

where $d_G(u) \le d$ *denotes the degree of* u *in* G*. The graph* G *is said to be* rapidly mixing *if, for every* $v_0 \in [k]$*, it holds that*

$$\frac{1}{2k} < \mathbf{Pr}_{(v_1,...,v_\ell)\leftarrow\mathcal{RW}_\ell}[v_\ell = v] < \frac{2}{k} \qquad (13)$$

As in the previous lecture, the key quantities in the analysis are the following probabilities that refer to the parity *of the length of a path obtained from the lazy random walk by omitting the self-loops* (transitions that remain at the current vertex). Let $p_0(v)$ (respectively, $p_1(v)$) denote the probability that a *lazy random walk of length* ℓ*, starting at* s*, reaches* v *while making an even* (respectively, *odd*) *number of real* (i.e., non-self-loop) *steps.* That is, for every $\sigma \in \{0,1\}$ and $v \in [k]$,

$$p_\sigma(v) \stackrel{\text{def}}{=} \mathbf{Pr}_{(v_1,...,v_\ell)\leftarrow\mathcal{RW}_\ell}[v_\ell = v \ \wedge \ |\{i \in [\ell] : v_i \neq v_{i-1}\}| \equiv \sigma \pmod 2] \qquad (14)$$

The path-parity of the walk $(v_1, ..., v_\ell)$ is defined as $|\{i \in [\ell] : v_i \neq v_{i-1}\}| \bmod 2$. By the rapid mixing assumption (for every $v \in [k]$), it holds that $\frac{1}{2k} < p_0(v) + p_1(v) < \frac{2}{k}$.

At this point we finally depart from the exposition of the previous lecture: Rather than considering the sum $\sum_{v\in[k]} p_0(v)p_1(v)$, we consider the sum $\sum_{\sigma\in\{0,1\}} \sum_{\{u,v\}\in E} p_\sigma(u)p_\sigma(v)$. If the sum is (relatively) "small", then we show that $[k]$ can be 2-partitioned so that there are relatively few

[25]The hypothesis $\bar{d} = \Omega(d)$ is used in Claim 13.3, where it is postulated that $m = \Omega(\sqrt{k/d\epsilon})$.

edges between vertices that are placed in the same side, which implies that G is close to being bipartite. Otherwise (i.e., when the sum is not "small"), we show that with significant probability, when Step 2 is started at vertex s, it is completed by rejecting G. These two cases are analyzed in the following two (corresponding) claims.

Claim 13.2 (a small sum implies closeness to being bipartite): *Suppose that* $\sum_\sigma \sum_{\{u,v\}\in E} p_\sigma(u)p_\sigma(v) \leq 0.01\epsilon\bar{d}/k$, *where* $\bar{d}$ *is the average degree of* $G = ([k], E)$. *Let* $V_1 \stackrel{\rm def}{=} \{v \in [k] : p_0(v) < p_1(v)\}$ *and* $V_2 = [k] \setminus V_1$. *Then, the number of edges with both end-points in the same* V_σ *is less than* $\epsilon\bar{d}k/2$.

Note that the proof of this claim is easier than the proof of the corresponding claim in the previous lecture.

Proof Sketch: Consider an edge $\{u, v\}$ such that both u and v are in the same V_σ, and assume, without loss of generality, that $\sigma = 1$. Then, by the (lower bound of the) *rapid mixing hypothesis*, both $p_1(v)$ and $p_1(u)$ are greater than $\frac{1}{2} \cdot \frac{1}{2k}$. Hence, the edge $\{u, v\}$ contributes at least $(1/4k)^2$ to the sum, and it follows that we can have at most $\frac{0.01\epsilon\bar{d}/k}{1/(16k^2)} < \epsilon\bar{d}k/2$ such edges. The claim follows. ■

Claim 13.3 (a large sum implies high rejection probability): *Suppose that* $\sum_\sigma \sum_{\{u,v\}\in E} p_\sigma(u)p_\sigma(v) \geq 0.01\epsilon\bar{d}/k$, *where* $\bar{d}$ *is the average degree of* $G = ([k], E)$, *and that Step 2 is started with vertex* s. *Then, for* $m = \Omega(\sqrt{k/\bar{d}\epsilon})$, *with probability at least* $2/3$, *there exist an edge with both endpoints in the same* R_σ (and rejection follows).

The proof of this claim is very similar to the proof of the corresponding claim in the previous lecture.[26]

The final result. Applying the reduction of Section 2.2, while approximating $\bar{d}$ and sampling edges by using adjacency queries (see Exercise 8), we obtain an alternative `Bipartiteness` tester, for the general graph model, that has expected time complexity $\text{poly}(\epsilon^{-1}\log k)\cdot(k/\bar{d})$.[27] Combining the two algorithms, we obtain.

Theorem 14 (testing `Bipatiteness` (in the general graph model), revised):[28] `Bipatiteness` *has a* (one-sided error) *tester of expected time* (and query) *complexity* $\text{poly}(\epsilon^{-1}\log k)\cdot\min(\sqrt{k}, k/\bar{d})$, *where* $\bar{d}$ *denotes the average degree of the input graph.*

In other words, ignoring $\text{poly}(\epsilon^{-1}\log k)$ factors, the time complexity of the tester is $O(\sqrt{k})$ if $\bar{d} \leq \sqrt{k}$ and $O(k/\bar{d})$ otherwise. We mention that the "non-smooth" behavior of the complexity

[26] Here we define $\zeta_{i,j} = 1$ if there exists an edge $\{u, v\} \in E$ such that the ℓth step of the ith walk reaches u, the ℓth step of the jth walk reaches v, and both walks have the same path-parity. Note that $\mathbb{E}[\zeta_{i,j}]$ equals the sum in the claim, since the events referring to different edges $\{u, v\}$ are mutually exclusive. We use the hypothesis that lower-bounds the said sum by $0.01\epsilon \cdot \bar{d}/k$, and the hypothesis that lower-bounds the number of pairs of walks by $\Omega(\epsilon^{-1}k/\bar{d})$.

[27] We believe that the poly$(\log k)$ factor can be eliminated when $\overline{\delta} \geq k^{\Omega(1)}$, since it is due to considerations related to the distribution of the endpoint of a random walk on regular k-vertex graphs. Recall that in the original context (of bounded-degree graphs), these graphs had constant degree, and so a random walk had to be of length $\Omega(\log k)$ in order to have its endpoint well distributed. But here we deal with $\bar{d}$-regular k-vertex graphs, where $\overline{\delta} > \sqrt{k}$, and so it stands to reason that a constant length random walk will do.

[28] We stress that this result refers to the testing model captured by Definition 2.

bound stated in Theorem 14 (i.e., the change of behavior at $\bar{d} \approx \sqrt{k}$) is not an artifact of its proof (which combines two different algorithms), but rather reflects the reality. *For every value of* $d \in [k]$*, any* `Bipartiteness` *tester in the general graph model must have query complexity* $\min(\sqrt{k}, k/d)$*, even when guaranteed that the input graph has average degree* $d \pm 1$.

5 Final notes

5.1 Gaps between the general graph model and the bounded-degree model

As argued at the begining of Section 2, a good starting point for the design of testers for the general graph model is the design of testers for the bounded-degree graph model. In Section 2 we presented cases in which either an adaptation of the latter testers or a local reduction (from testing in the general graph model) to testing in the bounded-degree graph model works. It is fair to indicate that there are cases in which such an adaptation inherently fails (and any reduction must have significant overhead). This is certainly the case when there are lower bounds on the complexity of testing graph properties in the general graph model that are significantly higher than the corresponding upper bounds that hold in the bounded-degree graph model. Examples include testing *cycle freeness* and *subgraph freeness*. In both cases (as well as for *degree regularity*), testers of time complexity $\text{poly}(1/\epsilon)$ are known for the bounded-degree graph model, but it is easy to see that testing these properties in the general graph model requires $\Omega(\sqrt{k})$ queries (even when the average degree is a constant).

Theorem 15 (lower bound on testing cycle freeness and subgraph freeness): *Testing the following properties in the general graph model requires* $\Omega(\sqrt{k})$ *queries, when allowed both incidence and adjaceny queries to a* k*-vertex graph.*

1. *Cycle-freeness.*
2. H*-freeness, for any fixed graph* H *having more than a single edge.*
3. *Degree regularity.*

Furthermore, this holds even if it is guaranteed that the average degree of the tested graph is between 1 and 2 (and the maximum degree is $\sqrt{k}$).

The furthermore clause clarifies that the difficulty lies in the varying vertex degrees (equiv., the gap between the average degree and the maximal degree) rather than in the magnitude of the average degree. (Part 3 is implicit in the proof of Proposition 6; in fact, we use the very same proof strategy here.)

Proof Sketch: We show that an algorithm that makes $o(\sqrt{k})$ queries cannot distinguish the following two distributions.

1. The uniform distribution on k-vertex graphs that consist of $k/2$ isolated edges.
2. The uniform distribution on k-vertex graphs that consist of $(k - \sqrt{k})/2$ isolated edges and a clique of $\sqrt{k}$ vertices.

The point is that as long as the algorithm makes no query to a vertex in the clique, the two distributions are identical. However, graphs in the first distribution are cycle-free and H-free (and degree regular), whereas graphs in the second distribution are $\Omega(1)$-far from being cycle-free (resp., H-free and degree regular). ■

Another lower bound. Theorem 15 asserts the existence of graph properties that are ϵ-testable with $\text{poly}(1/\epsilon)$ queries but requires $\Omega(\sqrt{k})$ queries for testing in the general graph model. Recall that the lower bound is established also under the guarantee that the average degree of the tested graph is $\Theta(1)$ and the maximum degree is $\sqrt{k}$, which represents a gap of $\Omega(\sqrt{k})$ between the average and maximal degrees. We mention that, in the general graph model, testing `triangle freeness` has query complexity $\Omega(k^{1/3})$ also when the average degree of the graph is $k^{1-o(1)}$, which represents a smaller gap between the average degree and the maximal degree [2].[29] This shows that *the query complexity of testing the graph property* Π *in the general graph model cannot be upper-bounded by* $\text{poly}(r(G), Q_{\texttt{dns}}, Q_{\texttt{bd}})$*, where* $r(G)$ *denotes the ratio between the maximal and average degrees in the tested graph* G*, and* $Q_{\texttt{dns}}$ (resp., $Q_{\texttt{bdg}}$) *denotes the query complexity of testing* Π *in the dense graph model* (resp., in the bounded-degree graph model).

5.2 History and credits

The study of property testing in the general graph model was initiated by Parnas and Ron [14], who only considered incidence queries, and extended by Kaufman, Krivelevich, and Ron [13], who considered both types of queries.[30] Needless to say, the aim of these works was to address the limitations of the previous models for testing graph properties; that is, to allow the consideration of arbitrary graphs. (Recall that the dense graph model is suitable mostly for dense graphs and the bounded-degree model is applicable only for graph of bounded degree.) Allowing the consideration of arbitrary graphs also strengthen the relation between property testing and standard algorithmic studies. However, forsaking the paradigm of representing graphs as functions means that the connection to the rest of property testing is a bit weakened (or at least becomes more cumbersome).

Turning to the specific results, we mention that the adaptation of the connectivity tester to the current model is due to [14]. The results regarding testing `Bipartitenss` in the general graph model were obtained by Kaufman, Krivelevich, and Ron [13]. This refers both to the upper and lower bounds when only incidence queries are allowed, and to the upper and lower bounds in the full fledged model (where also adjacency queries are allowed).

The lower and upper bounds on the complexity of degree estimation when only degree queries are allowed are due to Feige [7], and the corresponding bounds for the case when also incidence (and adjacency) queries are allowed are due to Goldreich and Ron [11]. The method presented in Section 3.2.1 is the one used in [11]; the alternative method presented in Section 3.2.2 was discovered recently by Eden, Ron, and Seshadhri [6].

[29]Note that this does not contradict Theorem 12, since the query complexity of ϵ-testing `triangle freeness` in the dense graph model is greater than any polynomial in $1/\epsilon$. Recall that Theorem 12 implies that if ϵ-testing `triangle freeness` in the dense graph model has query complexity $q(\epsilon)$, then ϵ-testing `triangle freeness` in the general graph model has query complexity $q(\rho\epsilon)$, where $\rho \cdot k$ is the average degree of the tested graph.

[30]The suggested treatement of extremely sparse graphs as captured in Definition 2 did not appear before (as far as we know).

5.3 Reflections

The bulk of algorithmic research regarding graphs refers to general graphs. Of special interest are graphs that are neither very dense nor have a bounded degree. In contrast, research in testing properties of graphs started (in [8]) with the study of dense graphs, proceeded to the study of bounded-degree graphs (in [9]), and reached general graphs only in [14, 13]. This evolution has historical reasons, which will be reviewed next.

Testing graph properties was initially conceived (by Goldreich, Goldwasser, and Ron [8]) as a special case of the framework of testing properties of functions. Thus, graphs had to be represented by functions, and two standard representations of graphs (indeed the ones underlying the dense graph model and the bounded-degree graph model) seemed most fitting in this context. In particular, in the dense graph model graphs are represented by their adjacency predicate, whereas in the bounded-degree (graph) model graphs are represented by their (bounded-degree) incidence functions. Hence, the representation of graphs by functions was maintained in the bounded-degree graph model, introduced by Goldreich and Ron [9], although the functions in this case were different. We stress that both models were formulated in a way that identifies the graphs with a specific functional representation, which in turn defines both the type of queries allowed to the tester and the notion of fractional distance (which underlies the performance guarantee).

The identification of graphs with a specific functional representation was abandoned by Parnas and Ron [14], who developed a more general model by decoupling the type of queries allowed to the tester from the distance measure: Whatever is the mechanism of accessing the graph, the distance between graphs is defined as the number of edges in their symmetric difference (rather than the number of different entries with respect to some specific functional representation). Furthermore, the relative distance is defined as the size of the symmetric difference divided by the actual (total) number of edges in both graphs (rather than divided by some (possibly non-tight) upper-bound on the latter quantity). Also, as advocated by Kaufman *et al.* [13], it is reasonable to allow the tester to perform both adjacency and incidence queries (and indeed each type of query may be useful in a different range of edge densities). Needless to say, this model seems adequate for the study of testing properties of arbitrary graphs, and it strictly generalizes the positive aspects of the two prior models (i.e., the models based on the adjacency matrix and bounded-degree incidence list representations).

We wish to advocate further study of the latter model. We believe that this model, which allows for a meaningful treatment of property testing of general graphs, is the one that is most relevant to computer science applications. Furthermore, it seems that designing testers in this model requires the development of algorithmic techniques that may be applicable also in other areas of algorithmic research. As an example, we mention that techniques in [13] underly the average degree approximation of [11]. (Likewise techniques of [9] underly the minimum spanning tree weight approximation of [4]; indeed, as noted next, the bounded-degree incidence list model is also more algorithmic oriented than the adjacency matrix model.)[31]

Let us focus on the algorithmic contents of property testing of graphs. Recall that, when ignoring a quadratic blow-up in the query complexity, property testing in the adjacency matrix representation reduces to sheer combinatorics (as reflected in the notion of canonical testers). Indeed, as shown in [12], a finer look (which does not allow for ignoring quadratic blow-ups in complexity) reveals the role of algorithmic design also in this model. Still, property testing in the

[31]Here and in the rest of this section, we use the terms “bounded-degree incidence list model” and “adjacency matrix model” rather than the terms “bounded-degree graph model” and “dense graph model” (used so far).

incidence list representation seems to require more sophisticated algorithms. Testers in the general graph models seem to require even more algorithmic ideas (cf. [13]).

To summarize, we advocate further study of the model of [14, 13] for two reasons. The first reason is that we believe in the greater relevance of this model to computer science applications. The second reason is that we believe in the greater potential of this model to have cross fertilization with other branches of algorithmic research. Nevertheless, this advocation is not meant to undermine the study of the dense graph and bounded-degree graph models. The latter models have their own merits and also offer a host of interesting open problems, which are of potential relevance to computer science at large.

5.4 Exercises

Exercise 1 (random bipartite graphs are good gadgets for the proof of Theorem 4): *Let $G = ((X,Y),E)$ be a random d-regular graph such that $|X| = |Y| = t$ and $E \subseteq \{\{x,y\} : x \in X \wedge y \in Y\}$.*

1. *Show that, with high probability, for every $S \subseteq X$ and $T \subseteq Y$ it holds that $|E(S,T)| = \Omega(d \cdot |S| \cdot |T|/t)$, where $E(S,T) = \{\{u,v\} \in E : x \in S \wedge y \in T\}$.*

2. *Using Part 1, show that, with high probability, for each 2-partition $(S,\overline{S})$ of the vertices of X such that $|S| \leq t/2$ and for every 2-partition $(T,\overline{T})$ of the vertices of Y it holds that $\min(|E(S,T)|, |E(\overline{S},\overline{T})|) = \Omega(d \cdot |S|)$.*

3. *Using Part 2, infer that any 2-partition of G that places $t' \leq t/2$ vertices of X on one side, has at least $\Omega(t'd)$ violating edges* (i.e., edges with both endpoints on the same side).

We mention that for a fixed set as in Part 3, a 2-partition of Y that has the least violating edges places each $y \in Y$ on opposite side to the majority of its neighbors.

Guideline: For Part 1, fix any S and T, and note that for a random d-regular $G = ((X,Y),E)$ it holds that $|E(S,T)| = \sum_{u \in S, v \in T} \zeta_{u,v}$, where $\zeta_{u,v}$ is a random variable indicating whether $\{u,v\} \in E$ (which means that $\mathbb{E}[\zeta_{u,v}] = d/t$). Establish a variant of Part 1 that refers to the case that the $\zeta_{u,v}$'s are totally independent, then handle the case that each vertex in X is assigned d random neighbors (while assuming, w.l.o.g., that $|S| \geq |T|$), and finally handle random d-regular graphs. The other parts follow easily.[32]

Exercise 2 (obtaining edge expanding bipartite graphs): *For any constants $d \in \mathbb{N}$ and $c > 1$, let $\{G_n = ([n], E_n)\}_{n \in \mathbb{N}}$ be a family of d-regular n-vertex expanding graphs in the sense that every $S \subset [n]$ of size at most $n/2$ it holds that $|\Gamma_n(S)| \geq c \cdot |S|$, where $\Gamma_n(S) = \cup_{v \in S}\{u \in [n] : \{u,v\} \in E_n\}$. Consider a bipartite graph B_n with vertex-set $[2n]$ such that $\{i, n+j\}$ is an edge in B_n if and only if either $\{i,j\} \in E_n$ or $i = j$. Prove that for every $S \subset [n]$ of size at most $n/2$ it holds that*

$$\sum_{y \in [n+1,2n]} \min(|\Gamma(y) \cap S|, |\Gamma(y) \setminus S|) \geq (c-1) \cdot |S|$$

where $\Gamma(y) \subseteq [n]$ denotes the set of neighbors of y.

[32] In Part 2, observe that if $|T| \geq t/2$ (resp., $|\overline{T}| \geq t/2$), then $d \cdot |S| \cdot |T|/t = \Omega(d \cdot |S|)$ (resp., $d \cdot |\overline{S}| \cdot |\overline{T}|/t = \Omega(d \cdot |\overline{S}|)$). In Part 3, let S denote the aforementioned t'-subset of X, and let T denote the set of vertices being on the same side as S.

Guideline: Observe that $\sum_{y\in[n+1,2n]} \min(|\Gamma(y)\cap S|, |\Gamma(y)\setminus S|)$ is lower-bounded by

$$|\{y \in [n+1,2n] : \Gamma(y)\cap S \neq \emptyset \ \wedge\ \Gamma(y)\setminus S \neq \emptyset\}| \ = \ |\Gamma_n(S)\cap\Gamma_n([n]\setminus S)|$$

which is at least $(c-1)\cdot|S|$, since $|\Gamma_n(S)| \geq c\cdot|S|$ and $|\Gamma_n([n]\setminus S)| \geq |[n]\setminus S|$.

Exercise 3 (distance preservation of the reduction presented in Section 2.2):[33] *Referring to the transformation presented in Section 2.2, suppose that G is transformed to G' by replacing vertices with c-edge expanding $\bar{d}$-regular bipartite graphs. Show that if G is ϵ-far from bipartite, then G' is $(c\cdot\epsilon/8)$-far from bipartite*

Guideline: Given a 2-coloring χ' of the vertices of G', consider a 2-coloring $\chi : [k] \to \{1,2\}$ obtained by coloring each $v \in [n]$ according to the majority color used by the exterenal vertices associated with v; that is, $\chi(v) = 1$ if the majority of the vertices in X_v (the external vertices of the bipartite graph replacing v) are χ'-colored 1, and $\chi(v) = 0$ otherwise. Denoting the minority vertices in X_v by S_v, observe that the number of χ'-monochromatic edges in the bipartite graph replacing v is at least $c\cdot\bar{d}\cdot|S_v|$, whereas the number of edges between S_v and other bipartite graphs is at most $\bar{d}\cdot|S_v|$. Hence, extending χ to the vertices of G' increases the number of monchromatic edges by a factor of at most $1/c$ (in comparison to χ'). It follows that the number of χ-monocromatic edges in G is at most $1/c$ times the number of χ'-monocromatic edges in G'. Recalling that G' has at most $4k\cdot 2\bar{d}/2$ edges (whereas G has $k\cdot\bar{d}/2$ edges), infer that if G' is δ-close to being bipartite, then G is $8\delta/c$-close to being bipartite.

Exercise 4 (on the uniform distribution on edges): *Show that selecting an edge uniformly at random in a given graph and selecting a random vertex with probability proportional to its degree in the graph are locally reducible to one another, where one of the reductions utilizes logarithmically many queries mainly in order to determine the degree of the sampled vertex.*

Exercise 5 (extending Proposition 6)[34] *For every $\rho \in (0,1)$, any constant-factor approximation algorithm for the average degree of a graph $G = ([k],E)$ must make $\Omega(k/\sqrt{|E|})$ queries to G, even when allowed degree, incidence and adjacency queries, and even if it is guaranteed that $|E| = \Theta(\rho k^2)$.*

Guideline: For $\rho > 1/k$ and $\gamma > 1$, proceed as in the proof of Proposition 6, while setting $k' = \sqrt{\gamma\rho}\cdot k$ (rather than $k' = \sqrt{\gamma k}$) and using $d \stackrel{\rm def}{=} \lfloor \rho k \rfloor$ matchings (rather than one).[35] Note that the average degree in the second distribution is d, whereas the average degree in the first distribution is $\frac{k-k'}{k}\cdot d + \frac{k'}{k}\cdot(k'-1) \approx (1+\gamma-\sqrt{\gamma\rho})\cdot d \geq (1+\gamma-o(\sqrt{\gamma}))\cdot d$, where we assume that $\rho = o(1)$. For $\rho < 1/k$, we also use $k' = \sqrt{\gamma\rho}\cdot k$ but only match ρk^2 of the remaining $k-k'$ vertices (rather than all of them).[36] Here, the average degree in the second distribution is $d \stackrel{\rm def}{=} \rho k$, whereas the average degree in the first distribution is $d + \frac{k'}{k}\cdot(k'-1) \approx (1+\gamma)\cdot d$.

[33] Based on a result in [13].

[34] Based on a result in [11].

[35] In the first distribution use d matchings of the remaining $k-k'$ vertices (rather than one), and in the second distribution just use d perfect matchings (rather than one).

[36] In the second distribution, we only match ρk^2 of the k vertices (rather than all of them).

Exercise 6 (getting rid of the need for a rough estimate – take 1): *Let $\nu : \{0,1\}^* \to (0,1]$ be a value functions and suppose that A is a deterministic algorithm for approximating ν when given a valid lower for it. Specifically, suppose that A satisfies the following conditions.*

1. A never overestimates ν: *For every x and b, it holds that $A(x,b) \le \nu(x)$.*
2. A performs well when given a valid lower bound on $\nu(x)$: *For some $\alpha \in (0,1)$ and every x, b, if $\nu(x) \ge b$, then $A(x,b) \ge \alpha \cdot \nu(x)$.*
3. The complexity of A grows linearly with $1/b$: *The complexity of A on (x,b) is upper bounded by $Q_x(b)$, where $Q_x : (0,1) \to \mathbb{N}$ satisfies $Q_x(b) = \Omega(1/b)$.*

Then, $\nu(x)$ can be approximate to within a factor of $1/\alpha$ within complexity $\widetilde{O}(Q_x(2^{-\lceil \log_2(\alpha\nu(x))^{-1}\rceil}))$. Actually, an upper bound of $\sum_{i\in[\lceil \log_2(1/\alpha\nu(x))\rceil]} Q_x(2^{-i})$ holds regardless of the growth rate of Q_x.

Guideline: On input x, invoke A iteratively such that in the i^{th} iteration A is invoked on input $(x, 2^{-i})$, and halt in iteration i if and only if the output is at least 2^{-i} (i.e., if $A(x,2^{-i}) \ge 2^{-i}$).[37]

Exercise 7 (getting rid of the need for a rough estimate – take 2): *In continuation to Exercise 6, suppose that A is randomized and that Conditions 1 and 2 only hold with probability $2/3$. Consider an algorithm as in the guidelines to Exercise 6, except that in the i^{th} iteration it invokes $A(2^{-i}, x)$ for $\Theta(\log Q_x(2^{-(i+1)}))$ times and treats the majority value as if it was the verdict of a deterministic algorithm. Analyze the probility that this algorithm outputs an $1/\alpha$-factor approximation of $\nu(x)$ as well as its expected complexity. Assuming that $Q_x(b/2) \le \text{poly}(Q_x(b))$ for every $b \in (0,1]$, upper-bound the expected complexity by $\widetilde{O}(Q_x(2^{-\lceil \log_2(1/\alpha\nu(x))\rceil}))$.*

Guideline: Observe that the probaility that A fails to output an $1/\alpha$-factor approximation of $\nu(x)$ is at most

$$\sum_{i\in[\lceil \log_2(1/\alpha\nu(x))\rceil]} \exp(-\Omega(\log Q_x(2^{-(i+1)}))) < \sum_{i\in[\lceil \log_2(1/\alpha\nu(x))\rceil]} 2^{-2(i+1)} < 1/8$$

and that its expected complexity is

$$\sum_{i\in[t]} O(Q_x(2^{-i}) \log Q_x(2^{-(i+1)})) + \sum_{i>t} 2^{-2\log Q_x(2^{-i}))} \cdot O(Q_x(2^{-i}) \log Q_x(2^{-(i+1)}))$$

where $t \overset{\text{def}}{=} \lceil \log_2(1/\alpha\nu(x))\rceil$. Lastly, note that $Q_x(b/2) \le \text{poly}(Q_x(b))$ implies that $\log Q_x(b/2) = O(\log Q_x(b))$.

Exercise 8 (estimating average degree and sampling edges by using adjacency queries):

1. *Show that the number of edges in a given graph $G = ([k], E)$ can be approximated to within any constant factor by making $O(k^2/|E|)$ adjacency queries, in expectation.*
2. *Show that given a graph $G = ([k], E)$, an edge can be sampled uniformly at random by making $O(k^2/|E|)$ adjacency queries, in expectation.*

[37]Note that in this case $\nu(x) \ge 2^{-i}$ (by Contition 1), and so Condition 2 is applicable.

Guideline: The key observation is that a random pair of vertices constitutes an edge with probability $|E|/\binom{k}{2}$. In Part 1, for any desired constant factor $\alpha > 1$, sample pairs till $t \stackrel{\rm def}{=} O((\alpha-1)^{-2})$ edges are seen, and output the empirical frequency (i.e., t over the number of trials).[38] In Part 2, apply the paradigm of repeated sampling.

References

[1] N. Alon. Testing subgraphs of large graphs. *Random Structures and Algorithms*, Vol. 21, pages 359–370, 2002.

[2] N. Alon, T. Kaufman, M. Krivelevich, and D. Ron. Testing triangle freeness in general graphs. In *17th SODA*, pages 279–288, 2006.

[3] I. Ben-Eliezer, T. Kaufman, M. Krivelevich, and D. Ron. Comparing the strength of query types in property testing: the case of testing k-colorability. In *19th SODA*, 2008.

[4] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. In *19th ICALP*, pages 190–200, 2001.

[5] A. Czumaj, O. Goldreich, D. Ron, C. Seshadhri, A. Shapira, and C. Sohler. Finding cycles and trees in sublinear time. *Random Structures and Algorithms*, Vol. 45(2), pages 139–184, 2014.

[6] T. Eden, D. Ron, and C. Seshadhri. Sublinear Time Estimation of Degree Distribution Moments: The Arboricity Connection. Manuscript, 2016.

[7] U. Feige. On sums of independent random variables with unbounded variance, and estimating the average degree in a graph. SIAM Journal on Computing, Vol. 35 (4), pages 964–984, 2006.

[8] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, pages 653–750, July 1998. Extended abstract in *37th FOCS*, 1996.

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

[10] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, Vol. 19 (3), pages 335–373, 1999. Extended abstract in *30th STOC*, 1998.

[11] O. Goldreich and D. Ron. Approximating Average Parameters of Graphs. *Random Structures and Algorithms*, Vol. 32 (3), pages 473–493, 2008.

[12] O. Goldreich and D. Ron. Algorithmic Aspects of Property Testing in the Dense Graphs Model. *SIAM Journal on Computing*, Vol. 40, No. 2, pages 376–445, 2011.

[13] T. Kaufman, M. Krivelevich, and D. Ron. Tight Bounds for Testing Bipartiteness in General Graphs. *SIAM Journal on Computing*, Vol. 33 (6), pages 1441–1483, 2004.

[38] In the analysis, letting $\rho \stackrel{\rm def}{=} 2|E|/k^2$, consider the probability that at least t (resp., at most t) edges are seen in a sample of size $(2-\alpha)\cdot t/\rho$ (resp., $\alpha \cdot t/\rho$).

[14] M. Parnas and D. Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, Vol. 20 (2), pages 165–183, 2002.