Property Testing*

Instructor: Asaf Shapira Scribed by Guy Rutenberg †

Spring 2015 ‡

1 Graph property testing

We say that a graph G on n vertices is *ε-far* from satisfying a property $\mathcal{P}$ if one needs to add or remove at least εn^2 edges in order to get a graph $G' \in \mathcal{P}$. For example, if $\mathcal{P}$ is the property of triangle-free graphs, then we need to remove edges. For induced C_4-free graphs we may need both remove and add edges.

We say a property $\mathcal{P}$ is *testable* if for every ε there exists $f(\varepsilon)$ such that one can distinguish between $G \in \mathcal{P}$ and G which is ε-far from $\mathcal{P}$ using $f(\varepsilon)$ queries on the graph.

Fact 1.1. *For every graph H, the propery of H-freeness is testable.*

Lemma 1.1 (Removal Lemma)**.** *Suppose G is ε-far from being H-free then G contains $f(\varepsilon, H)n^h$ copies of H (where $h = |V(H)|$).*

The Removal Lemma gives us a good test for H-freeness, because it implies a constant probability (not depending on n) that h random vertices form a copy of H. This check can be repeated $1/f(\varepsilon, H)$ times to get a probability $\geq 1/2$ to distinguish correctly.

Exercise 1.1. Prove in a simple way that if G is ε-far from being H-free, then G contains $\varepsilon n^2/h^2$ edge disjoint copies of H.

Conjecture 1.1 (Erdős)**.** Suppose G is ε-far from being k-colorable, then G contains a subgraph of size $f(k, \varepsilon)$ which is not k-colorable.

The existence of $f(2, \varepsilon)$ was proved in by Bollobas-Erdős-Simonovits-Szemerédy (1978) and for general k by Rödl-Duke (1985). Both proofs used the Regularity Lemma, and as a result the bounds are very large, however it still proves that k-colorability is a testable property. Komlós (1997) showed that $f(2, \varepsilon) = \Theta(\sqrt{1/\varepsilon})$, by showing that if G is ε-far from being 2-colorable then G must contain an odd cycle of length $O(\sqrt{1/\varepsilon})$. To show that Komlós' bound is tight consider a blowup of an odd $\sqrt{1/\varepsilon}$ cycle.

*Part of a course on Topics in Combinatorics.

†guyrutenberg@mail.tau.ac.il

‡Typeset on June 29, 2015.

Exercise 1.2. Show that $f(2,\varepsilon) = O(1/\varepsilon)$. (Hint: Consider a minimal odd length cycle and count the number of edges to any other vertex and remove them, repeat the process).

Exercise 1.3. Show that Komlós' bound of $O(\sqrt{1/\varepsilon})$ mentioned above is tight. (Hint: Consider a blowup of an appropriate odd cycle.)

Exercise 1.4. Show that there is a graph which is ε-far from being bipartite, and such that, with probability at least $2/3$, a random sample of $\frac{1}{100\varepsilon}$ vertices does *not* span an odd cycle. (Hint: Consider an appropriate blow-up of C_3.)

Goldreich, Goldwasser and Ron (1996) obtained a significant improvement, by showing that a sample of size $O(1/\epsilon^3)$ suffices. This was later improved by Alon-Krivelevich (2002) to $O(1/\epsilon^2)$ (with $\tilde{O}(1/\epsilon)$ for $k = 2$) and recently Sohler (2014) obtained a near tight $\tilde{O}(1/\epsilon)$ bound for every fixed k. We give the argument of Goldreich, Goldwasser and Ron for $k = 2$.

Theorem 1.1. *If G is ε-far from being bipartite, then a random vertex-set of size $O(\log(1/\varepsilon)/\varepsilon^2)$ spans a non-bipartite subgraph with probability $\geq 1/2$.*

The proof will follow from the next two lemmas. Let us say that a set S is *good* if all but at most $\epsilon n/10$ of the vertices of degree at least $\epsilon n/10$ have at least one neighbor in S.

Lemma 1.2. *If S is a random sample of vertices of size $\frac{10\log(100/\varepsilon)}{\varepsilon}$ (allowing repetitions), then with probability $\geq 9/10$ S is good.*

Lemma 1.3. *Suppose S is a good vertex set of size at most $\frac{10\log(100/\varepsilon)}{\varepsilon}$. Then with probability $\geq 9/10$ a random set T of $\frac{20\log(200/\varepsilon)}{\varepsilon^2}$ pairs satisfies $G[S \cup T]$ not a bipartite graph.*

Proof (Theorem 1.1). Lemma 1.2 and Lemma 1.3 imply Theorem 1.1: Sample $\frac{10\log(100/\varepsilon)}{\varepsilon}$ vertices, denote those by S, and another $\frac{40\log(200/\varepsilon)}{\varepsilon^2}$, denote those T. Let A be the even that S is good, B is the event that $S \cup T$ is not bipartite. By the lemmas we have $\Pr[A] \geq 9/10$ and $\Pr[B \mid A] \geq 9/10$. Hence

$$\Pr[B] \geq \Pr[B \mid A] \Pr[A] \geq 8/10.$$

□

Proof (Lemma 1.2). Suppose a vertex v has degree at least $\varepsilon n/10$. The probability that S does not contain any neighbor of v is at most

$$(1-\varepsilon/10)^{|S|} \leq (1-\varepsilon/10)^{\frac{10\log(100/\varepsilon)}{\varepsilon}} \leq e^{-\log(100/\varepsilon)} \leq \varepsilon/100$$

Therefore, the expected number of vertices with degree at least $\varepsilon n/10$ and without any neighbor in S is at most $\varepsilon n/100$. By Markov's inequality we thus get that there are more than $\varepsilon n/10$ such vertices is less than $1/10$. □

Proof (Lemma 1.3). Suppose S is a good set of at most $10\log(100/\epsilon)/\epsilon$ vertices and fix a 2-coloring, $c : S \to \{0, 1\}$, of the graph spanned by S (not necessarily a valid coloring). Let H be the set of vertices with at least one neighbor in S. For each vertex $x \in H$ pick one of its neighbors in S and assign x the opposite color.

We claim that this coloring induces at least $\varepsilon n^2/2$ edges within H which are improperly colored. To see this, suppose otherwise and remove from G any edge incident with a vertex not in H. There are at most $\varepsilon n^2/10$ edges incident with a vertex of degree $\leq \varepsilon n/10$ (true for every graph). There are $\varepsilon n^2/10$ edges incident with a vertex of degree $\geq \varepsilon n/10$ not in H (because S is good). There are $\leq |S|n \leq \varepsilon n^2/10$ edges incident with S. Additionally, remove the improperly colored edges in H. In total, we removed less than εn^2 edges and got a 2-colorable graph, a contradiction to the ε-farness of G.

We conclude that for every 2-coloring c there are at least $\varepsilon n^2/2$ pairs of vertices $\{x, y\}$ such that it is impossible to extend c to a valid coloring of $S \cup \{x, y\}$. Therefore, a random pair from G "invalidates" c with probability $\varepsilon/2$. Because T is made up of $\frac{20\log(200/\varepsilon)}{\varepsilon^2}$ pairs, the probability that c can be extended to a valid coloring of $S \cup T$ is at most

$$(1 - \varepsilon/2)^{\frac{20\log(200/\varepsilon)}{\varepsilon^2}} \leq e^{-10\log(200/\varepsilon)} = \left(\frac{\varepsilon}{200}\right)^{\frac{10}{\varepsilon}}$$

Namely, for a fixed c, the probablity that c can be extended to cover $S \cup T$ entirely is less than $(\varepsilon/200)^{(10/\varepsilon)}$. Because there are only $2^{|S|} \leq (100/\varepsilon)^{10/\varepsilon}$ 2-colorings of S, we get (using a union bound) that with high probability $S \cup T$ is not bipartite. □

2 Testing H-freeness

The removal lemma implies that for any fixed H, the property of being H-free is testable with $1/f(\varepsilon, h)$ queries. Let us say that H is *easily testable* if $f(\varepsilon, H) \leq poly(1/\varepsilon)$. We will prove the following precise characterization.

Theorem 2.1 (Alon (2001))**.** *H is easily testable if and only if H is bipartite.*

We first show that to prove the above it is enough to consider graphs H of a certain type. We write $f : H \mapsto K$ to denote a homomorphism from H to K, that is $(x, y) \in E(H) \to (f(x), f(y)) \in E(K)$.

Lemma 2.1 (Erdős-Simonovits (1983))**.** *The following holds for every $k, t, \varepsilon > 0$ and $n \geq n_0(k, t, \varepsilon)$: If a k-uniform hypergraph contains εn^k edges, then it contains $C\varepsilon^{t^k} n^{kt}$ copies of the complete k-partite k-graph with each class of size t, for some $C = C(k, t)$.*

Exercise 2.1. Suppose that there is a homomorphism $H \mapsto K$, and K is a subgraph of H.

1. If K is easily testable then so is H.
2. The removal lemma and Erdős-Simonovits imply that if K is not easily testable, then so is H.
3. Suppose F is an m-vertex graph which is ε-far from being K-free and contains only $f(\varepsilon)m^k$ copies of K (where $k = |V(K)|$). Then if G is a n/m-blowup of K then G is ε/h^2-far from being H-free and contains at most $f(\varepsilon)n^h + n^h/m$ copies of H. Actually, note that it is enought to show that for every ε there is $m \geq \text{superpoly}(1/\varepsilon)$ for which it is possible to construct a k-vertex graph which is ε-far from being K-free and contains a few copies of K.

A *core* of a graph H is a subgraph K so that there is some $f : H \mapsto K$ but there is no $f' : H \mapsto K'$ for every $K' \subset K$. We say that H is a core if it is its only core[1].

Exercise 2.2. Show that if H is a core then every $f : H \mapsto H$ is an isomorphism.

Exercise 2.3. Show that all the cores of a graph are isomorphic.

It is enough to prove Theorem 2.1 when H is a core. Because $K_{1,1}$ (an edge) is easily testable (convince yourself) and because the core of every bipartite graph is an edge, we infer that every bipartite graph is easily testable. This proves the "if part" of Theorem 2.1.

Since the core of a non-bipartite graph is not bipartite (right?), it is sufficient to show that every non-bipartite core is not easily testable. Furthermore, it is sufficient to show (3). So our current goal is to suppose that H is a non-bipartite core and given ε find $m \geq \text{superpoly}(1/\varepsilon)$ and construct a m-vertex graph G such that G is ε-far from being H-free and G contains $\frac{m^h}{\text{superpoly}(1/\varepsilon)}$ copies of H.

We will need the following well known fact (which appeared in the previous course):

[1]Note that we are not claiming (yet) that the core is unique. Actually, H might have many cores, but we will soon prove that they are all isomorphic.

Fact 2.1 (Behrend (1964)). *For every $r \geq 2$ there and m there is an $X \subseteq [m]$ of size at least $m/e^{10\sqrt{\log m \log r}}$ that does not contain a (non-trivial) solution to the equation*

$$x_1 + x_2 + \cdots + x_r = r x_{r+1} \tag{2.1}$$

We now conclude the proof of the "only if" part of Theorem 2.1. Since H is not bipartite it has an odd cycle. Rename the vertices of $V(H)$ using $\{1, \ldots, h\}$ such that $1, \ldots, r+1, 1$ is an odd cycle in H (in this order!). Given $\varepsilon > 0$ let m be the largest integer satisfying

$$m/e^{10\sqrt{\log m \log r}} \geq \epsilon m \ .$$

It is easy to check that

$$m \geq (1/\epsilon)^{c \log 1/\epsilon}$$

for some $c = c(r)$. Let $S \subseteq [m]$ be a subset satisfying (2.1) of size at least ϵm.

We define G to be an h-partite graph on h vertex sets $U_1, \ldots, U_h$ as follows: We think of U_i as the set of integers $1, \ldots, i \cdot m$. Then, for every $x \in [m]$ and every $s \in S$ and $s \in S$ we put a copy of H on vertices $x \in U_1, x+s \in U_2, \ldots, x+(h-1)s \in U_h$ and where the vertex chosen from U_i plays the role of vertex $i \in V(H)$. That is, we connect (v_i, v_j) iff $(i,j) \in E(H)$.

Observe that we placed εm^2 copies of H.

Exercise 2.4. Those copies are edge-disjoint (exercise) and in particular εm^2 edges need to be removed to turn G into H-fre graph. Therefore, G is ε/h^4-far from H-free.

It remains to prove that G contains few copies of H. We will actually prove that G contains at most $m^2 m^{h-(r+1)}$ copies of H. Since $r+1 \geq 3$ this is at most $m^{h-1} = \frac{m^h}{m}$.

1. Every copy of H in G contains a C_{r+1} going "through" the sets $U_1, \ldots, U_{r+1}$ (in this order).

2. G contains at most m^2 cycles C_{r+1} passing through the sets $U_1, \ldots, U_{r+1}$.

First note that the above two claims imply that G contains at most $m^2 m^{h-(r+1)}$ copies of H. To prove (1), observe that every copy of H, spanned by vertices $v_i \in U_{f(i)}$ defines a natural homomorphism $f : H \mapsto H$ which sends i to $f(i)$. Then (1) follows from Exercise 2.2 because if f is an isomorphism then a cycle of length $r+1$ is mapped to a cycle of length $r+1$ on the vertices $1, \ldots, r+1$ and hence H contains a C_{r+1} passing through $U_1, \ldots, U_{r+1}$.

To prove (2), assume that $x_1 \in U_1, \ldots, x_{r+1} \in U_{r+1}$ is one such C_{r+1}. Observe that the way we defined G means that $x_{i+1} - x_i = s_i \in S$ for every $1 \leq i \leq r-1$ and $x_{r+1} - x_1 = r s_{r+1}$ for some $s_r \in S$. This means that

$$x_1 + s_1 + s_2 + \cdots + s_r - r s_{r+1} = x_1 \ ,$$

that is, that $s_1 + s_2 + \cdots + s_r = r s_{r+1}$. But the fact that S contains no non-trivial solution to such an equation means that $s_1 = s_2 = \ldots = s_{r+1}$ implying that G contains precisely $m|S|$ such copies of C_{r+1}.

3 Property Testing for Sparse Graphs

Suppose G is an n vertex graph of maximum degree d. We will think of d as fixed and n large. We say that G is ε-far from satisfying $\mathcal{P}$ if one should add/delete at least εn edges in order to turn G into a graph satisfying $\mathcal{P}$. We will focus here on monotone properties, that is, properties closed under vertex/edge removal, so being far means that one should remove εn edges to make G satisfy $\mathcal{P}$. In the setting of sparse graphs, what the algorithm can do, is sample a vertex and look at its neighborhood. Note that while in dense graph we ask what can we learn from looking at randomly selected induced subgraphs of bounded size (i.e., we pick a set S of size $f(\varepsilon)$ and look at $G[S]$) in the setting of sparse graphs the question is what can we learn from randomly picking $f(\varepsilon)$ vertices and looking at the balls of radius $g(\varepsilon)$ around them.

We say that property $\mathcal{P}$ is testable if there is a function $f(\varepsilon, d)$ and an algorithm that does the following: Given a graph G the algorithm can randomly/deterministically asks for the neighborhood of at most $q = f(\varepsilon, d)$ vertices. The questions are allowed to be adaptive. Using this information the algorithm should distinguish, with probability at least $2/3$, between the case that G satisfies $\mathcal{P}$ from the case that G is ε-far from $\mathcal{P}$. Recall that we assume that G has bounded degree d for some fixed d.

Recall that H-freeness is testable in dense graphs. It turns out that for sparse graphs the analogous question is very easy.

Exercise 3.1. Show that H-freeness is testable in sparse graphs.

Recall that in dense graphs k-colorability is testable. It turns out that in sparse graphs the situation is very different. Erdős proved that there are bounded-degree graphs that are ε-for from being (say) 10-colorable yet every induced subgraph on $c(\varepsilon)n$ vertices is 3-colorable. Hence we cannot hope to test 10-colorability with $o(n)$ queries, at least not with one-sided error, that is, when we want to always accept graphs that are 10-colorable. Note that intuitively, in order to test a property with one-sided error we need to show that if G is far from satisfying $\mathcal{P}$ then we can find a "proof" of this fact using few queries (with probability at least $2/3$). It turns out that even when allowing two-sided error, 3-colorability cannot be tested in sparse graphs with $o(n)$ queries (note that with $O(n)$ queries we have the entire graph).

Let us focus from now on testing planarity. Since any monotone graph property is testable in dense graphs, we infer that planarity is testable. Actually, this is a simple consequence of the fact that a planar graph contains at most $3n - 6$ edges.

Exercise 3.2. Show that planarity is (easily) testable in dense graphs.

We now recall the so called Planar Separator Theorem.

Theorem (Lipton-Tarjan (1980))**.** *Every planar graph G contains a vertex set of size at most $10\sqrt{n}$ whose removal disconnects G into connected components of size at most $2n/3$.*

This can be used to prove the following:

Exercise 3.3. Show that there are bounded degree graphs that are ε-far from being planar, yet any set of $f(\varepsilon)$ vertices spans a planar graph. In other words, planarity is not testable with one-sided error in sparse graphs.

Our aim now is to prove the following somewhat surprising fact:

Theorem 3.1 (Benjamini-Schramm-Shapira (2010))**.** *Planarity is testable in sparse graphs.*

In other words, suppose we take two graphs G and G' both of maximum degree d, where G is planar and G' is ε-far from being planar. Let D be the distribution obtained by picking a random vertex v from G and looking at a ball of radius $f(\varepsilon, d)$ around v. Let D' be the distribution obtained by the same process with respect to D'. Then D and D' have statistical difference at least $c(\varepsilon)$.

The first step in the proof of Theorem 3.1 is the following. A partition of $V(G)$ into sets $\{V_1, \ldots, V_p\}$ is called an (ε, k)-partition if each V_i spans a connected subgraph of G of size at most k, and at most εn edges of G connect distinct V_i, V_j. In what follows we will use $C_1, C_2, \ldots$ to denote constants that depend on d but are independent of n and ε.

Exercise 3.4. Use the separator theorem to prove that for every d there is C_1 so that any planar graph of maximum degree d has an $(\varepsilon, C_1/\varepsilon^2)$-partition. (Hint: Divide an conquer)

Let us say that a set of vertices S is an (ε, k)-set if $|S| \leq k$ and there are at most $\varepsilon|S|$ edges connecting S to $V \setminus S$.

Exercise 3.5. Show that if G is a planar graph of maximum degree d, then a randomly chosen vertex belongs to some $(\varepsilon/4, C_2/\varepsilon^4)$-set with probability at least $1 - \varepsilon/40d$.

We now prove Theorem 3.1 based on the proof of Hasidim-Kelner-Nguyen-Onak (2009).

Consider the following greedy partition algorithm for constructing a partition P of $V(G)$:

Algorithm 1 Random Greedy Partition Algorithm

Input: ε, d and a graph G of maximum degree d
Output: Partition P of $V(G)$

1. Pick a random permutation $\sigma \in S_n$
2. Iteratively pick the "first" among the remaining vertices and look[2] for an $(\varepsilon/4, C_2/\varepsilon^4)$-set containing v. If such a set S is found then add S to P, remove S from the graph and continue. If not, then add $\{v\}$ to P, remove v from the graph and continue.

[2]Here we "fix" some way of taking a vertex v and looking for an $(\varepsilon/4, C_2/\varepsilon^4)$-set containing it. In what follows when we refer to such a procedure we will mean the same procedure fixed here.

Note that for *any* graph G of bounded degree d, the algorithm constructs a partition with clusters of size at most C_2/ε^4. For what follows, let us say that a vertex v in a cluster of P is a "leader" if v is the first vertex of the cluster according to σ.

Exercise 3.6. Show that if G is a planar graph of maximum degree d, then with probability at least $9/10$ (over the choice of σ), the greedy algorithm produces an $(\varepsilon/2, C_2/\varepsilon^4)$-partition. (Hint: Use the previous exercise)

Theorem 3.2 (Partiton Oracle)**.** *There is an algorithm that given ε, d and a graph G of bounded degree d does/satisfies the following:*

1. *The algorithm (implicitly) picks a random permutation $\sigma \in S_n$ which (implicitly) determines a partition $P = P(\sigma)$ of $V(G)$.*
2. *For any σ, the partition P consists of clusters of size at most C_2/ε^4.*
3. *Given a vertex v, the algorithm returns the connected component of P to which v belongs by making $2^{2^{C_4/\varepsilon^4}}$ queries to G.*
4. *If G is planar, then with probability at least $4/5$, $P(\sigma)$ is a $(\varepsilon, C_2/\varepsilon^4)$-partition.*

Exercise 3.7. Use the above theorem together with Exercise 3.4 to deduce that planarity is testable (Theorem 3.1).

Proof (Theorem 3.2). Set $k = C_2/\varepsilon^4$. Let us pick a random permutation of $V(G)$ by assigning each vertex a random number $r(v) \in [0, 1]$. Given a permutation σ, let P' be the partition resulting from running the greedy partition algorithm described above. For each vertex v let $P'[v]$ be the cluster of vertices to which v belongs to in P'. We now describe a recursive "local" algorithm for finding $P'[v]$. Let us first assume that v is a leader. Observe (crucially) that in order to compute $P'[v]$ we first need to know which vertices at distance at most k from v belong to some other cluster, whose leader w satisfies $r(w) < r(v)$. Hence, in order to properly compute $P'[v]$ it is enough[3] to first recursively compute $P[w']$ for all w' within distance at most $2k$ from v that satisfy $r(w') < r(v)$. Once we compute all these values, we can look for an $(\varepsilon/2, k)$-set as in the greedy algorithm. Now, if v is not a leader, then $P'[v] = P'[w]$ where w is the leader of the cluster to which v belongs. Hence in this case it is enough to compute $P'[w]$ for all w at distance at most k satisfying $r(w) < r(v)$. We conclude, that for all v, in order to find $P'[v]$ it is enough to recursively compute $P'[w]$ for w within distance $2k$ from v which satisfy $r(w) < r(v)$. If $v \in P'[w]$ for one of these $w's$ then we know $P'[v]$, otherwise, we look for an $(\varepsilon/2, k)$-set as in the greedy algorithm.

Note that the recursive local algorithm for finding $P'[v]$ might potentially compute $P'[w]$ for all vertices of G. Let us then compute the expected number of recursive calls made by the algorithm when computing $P'[v]$. Consider the recursion tree T of the execution of the above local algorithm. Since computing $P'[v]$ invokes recursive calls

[3]Note that it is enough to first compute $P[w]$ for *all* w within distance k from v but that approach would turn out to be inefficient.

only for vertices at distance at most $2k$ from v we conclude that T has degree at most d^{2k}. In particular, for each t there are at most $(d^{2k})^t$ vertices in T at distance t from the root v. Now note that in order for the algorithm to reach a vertex x_t at depth t in T, connected to v by a path $v = x_1, x_2, \ldots, x_t$ we must have $r(x_1) > r(x_2) > \cdots > r(x_t)$. Since the probability of this event is $1/t!$, we infer that the expected number of recursive calls the algorithm makes when computing $P'[v]$ is at most

$$\sum_{t=1}^{\infty} \frac{(d^{2k})^t}{t!} = e^{d^{2k}} = e^{d^{2C_2/\varepsilon^4}} .$$

By Markov's Inequality, the probability of making more than $\frac{20d}{\varepsilon} \cdot e^{d^{2C_2/\varepsilon^4}} \leq 2^{2^{C_3/\varepsilon^4}}$ recursive calls when computing $P'[v]$ is at most $\varepsilon/20d$.

Let us say that a vertex v is *bad* if computing $P'[v]$ requires more than $2^{2^{C_3/\varepsilon^4}}$ recursive calls. We conclude that the expected number of bad vertices is at most $\varepsilon n/20d$. By Markov's Inequality, the number of bad vertices does not exceed $\varepsilon n/2d$ with probability at least $9/10$.

Let us now define the partition P and the algorithm in the statement of Theorem 3.2. Given vertex v the algorithm runs the local algorithm described above for computing $P'[w]$ for all vertices w within distance at most k from v with the restriction that whenever we try to compute $P'[w]$ we stop the execution after $2^{2^{C_3/\varepsilon^4}}$ recursive calls. If one of these recursive calls found a cluster containing v we return this cluster; otherwise, we return $\{v\}$ (that is, v will be in a cluster of size 1). Let P be the resulting partition. Observe that if $P'[v] = P'[w] = S$ then the above algorithm will either return S when the input is either v,w or will return $\{v\}$,$\{w\}$ in these cases, respectively. This means that P is well defined. Note that for each cluster $S \in P'$ either $S \in P$ or P breaks S into singletons. Also, the latter happens if and only if *all* vertices in S are bad.

Note that since σ determines P' as well as the bad vertices, it also determines P. This justifies item (1) of Theorem 3.2. Since P refines P' and all clusters of P' have size at most k we get item (2). As to item (3), the algorithm above clearly runs in time at most $d^k 2^{2^{C_3/\varepsilon^4}} \leq 2^{2^{C_4/\varepsilon^4}}$. We now prove item (4). First, observe that by Exercise 3.6, with probability at least $9/10$ the partition P' has at most $\varepsilon n/2$ edges connecting different clusters. Hence, it is enough to show that with probability at least $9/10$ partition P has no more than $\varepsilon n/2$ additional edges connecting different clusters. Since P breaks a cluster S of P' into singletons only if all vertices of S were bad, we conclude that the number of additional edges connecting different clusters of P is at most d times the number of bad vertices. Since we know that with probability at least $9/10$ the number of bad vertices does not exceed $\varepsilon n/2d$ we are done.

Now the above discussion assumed that we first picked a random permutation σ by assigning a random number $r(v)$ to all vertices. Since we cannot allow this if we want to implement the algorithms queries in constant time, we instead pick the the random number $r(v)$ "on the fly". That is, each vertex is assigned a random number $r(v) \in [0, 1]$ the moment we *first* encounter v. This value is then kept for in case this vertex is visited again, either when computing the answer to the same query $P[v]$ or when answering

another query $P[v']$. □

The best result on testing planarity is by Levi-Ron who designed a version of Theorem 3.2 where each query is answered by making $(1/\varepsilon)^{C \log 1/\varepsilon}$ queries to the graph.