6.889 Sublinear Time Algorithms February 27, 2019

Lecture 7

Lecturer: Ronitt Rubinfeld *Scribe: Aaron Berger*

1 Outline

Today we discussed an algorithm for testing whether a graph is bipartite.

2 The Tester

In the *adjacency matrix model* we have a graph $G = (V, E)$ represented by an adjacency matrix A_{ij}, where $A_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise. We assume we can query any $A_{i,j}$ in one step.

Definition 1 *In this model, say G is ϵ-far from property P if we must change more than ϵn^2 entries in $A(G)$ (i.e. ϵn^2 edges) to make G satisfy property P.*

Note that under this model, some properties become trivial. For example, every graph can be made connected by adding at most $n - 1$ edges (just add a spanning tree). Then since $\epsilon n^2 > n - 1$ for sufficiently large n, no sufficiently large graphs are ϵ-far from being connected. An algorithm to test this would therefore run in constant time (as it would just pass every graph)!

The property we want to test today will be bipartiteness.

Definition 2 *The following equivalent definitions characterize bipartiteness.*

1. *One can color each node of V either red or blue such that there is no monochromatic edge (i.e., no edge with both vertices having the same color.)*
2. *One can partition V into two parts (V_1, V_2) such that no edge has both vertices in V_1 or both vertices in V_2.*

Consequently, we obtain the following two equivalent characterizations of being ϵ-far from bipartite:

Proposition 1 *A graph is ϵ-far from bipartite if and only if one of the following (equivalent) properties holds:*

3. *One must remove more than ϵn^2 edges to satisfy property (1) above, i.e., for the graph to have a proper 2-coloring.*
4. *One must remove more than ϵn^2 edges to satisfy property (2) above, i.e., for the graph to have a partition $V = V_1 \sqcup V_2$ with no edge having both vertices in V_1 or V_2.*

Our first attempt at producing an algorithm is as follows:

Algorithm 1: First Attempt Bipartite Tester

Input : $G = (V, E)$ graph in adjacency matrix form
Output: Tests whether G is bipartite or ϵ-far from bipartite

Pick $m = \Theta(?)$ random pairs (i, j) and query $A_{i,j}$.
For all partitions $V_1 \sqcup V_2 = V$:
    violating$_{v_1,v_2} \leftarrow$ # edges in sample lying entirely in V_1 or V_2.
If all violating$_{v_1,v_2} > 0$, `FAIL`
Else, `PASS`

If G is bipartite, there is some partition with no violating edges (characterization 2) and therefore we will always `PASS`. If G is ϵ-far from being bipartite, then every partition has at least ϵn^2 violating edges (so an ϵ-fraction of its edges are violating). The property that we see a violating edge for a fixed partition is therefore at least $1-(1-\epsilon)^m$, because we chose m edges randomly and each is good with probability at most $(1-\epsilon)$.

Then since there are 2^n partitions, the probability that none of them pass can be (union) bounded by $1-(2^n(1-\epsilon)^n)$, so in order to make this less than 1 we'd need $m=\Omega(n/\epsilon)$. Plus we'd still need to check each of the 2^n partitions. So we use a bunch of queries and check a bunch of partitions, so both are inefficient steps that we'd like to improve.

The plan to fix this will be to use an *ϵ-net*, i.e. rather than query each of the 2^n partitions, we will query a small number of partitions so that we don't get every partition but we at least get something "close" to every partition. Hopefully the number of violating edges will be similar for partitions that are close, so we can get information about violating edges for all partitions by only querying a small number. Here is how that might look:

Algorithm 2: Ideal Bipartite Tester

Input : $G=(V,E)$ graph in adjacency matrix form
Output:

1 Pick $m=\Theta(\frac{1}{\epsilon^2}\log(1/\epsilon))$ random nodes.
2 If the induced subgraph is not bipartite, `FAIL`
3 Else, `PASS`

Every bipartite graph will still pass this algorithm (think of the coloring definition to see this, for example). It's not at all clear that a graph that is far from being bipartite will fail, however. Here's a second similar algorithm which should convince us that the first one works.

Algorithm 3: Actual Bipartite Tester

Input : $G=(V,E)$ graph in adjacency matrix form
Output:

1 Randomly pick $\Theta(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ and call this set U.
2 Randomly pick $\Theta(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$ pairs of nodes and call this set $U'=\{(u_1,v_1),(u_2,v_2),\ldots\}$.
3 If U is not bipartite, `FAIL`.
4 Else, **for each** bipartition $U=U_1\sqcup U_2$ (with no violating edges):
5 Attempt to partition all vertices of the graph into two sets $W_1\sqcup W_2$, where W_1 are the vertices connected to something in U_2, and W_2 are the vertices connected to something in U_1, or to neither.
6 If this attempt fails, i.e., something is connected to both, then the partition of U is bad; **continue** to the next one.
7 Count the edges in U' that violate the partition W_1, W_2, i.e., those with both endpoints entirely in one part or the other. If this number is less than $\frac{3}{4}\epsilon|U'|$, then `PASS`.
8 Else **continue** to the next partition of U.
9 If no partition of U succeeds, then `Fail`.

In this algorithm we have the opposite problem to usual: here it's easy to show that bad graphs fail, but hard to show that good graphs pass. If G is ϵ-far from bipartite, then at least an ϵ-fraction of the edges will violate any choice of W_1, W_2 (Property 4). So with very high probability (Chernoff bound) our random choice of U' will have more than a $\frac{3}{4}\epsilon$ fraction of its edges be violating, every time.

On the other hand, say G was actually bipartite. Then it has at least one bipartition $V=(Y_1,Y_2)$ with no violating edges. For any sample U, let's consider the step of the algorithm when U_1 is set to $Y_1\cap U$ and U_2 is set to $Y_2\cap U$ (this has no violating edges so it will be tested at some point, unless we've already passed.) Let $W_1^{u_1,u_2}, W_2^{u_1,u_2}$ be what the algorithm chooses for W_1 and W_2 at this iteration. We want to show that these W_1 and W_2 are a good approximation for Y_1 and Y_2. If a vertex is connected to

$W_1^{u_1,u_2} \subset Y_1$, then it must be in Y_2 and also must be put in $W_2^{u_1,u_2}$. Similarly, if a vertex is connected to $W_2^{u_1,u_2} \subset Y_2$ then it must be in Y_1 and also must be put in W_1. Therefore the only vertices v where Y_1 and W_1, or Y_2 and W_2 differ, are those vertices that are connected to nothing in U whatsoever. There are two cases for such a vertex v that has no neighbor in U:

- small degree: $\deg(v) < \frac{\epsilon}{4}n$. Call this set of vertices A
- high degree: $\deg(v) \geq \frac{\epsilon}{4}n$. Call this set of vertices B.

Now a violating edge must have at least one vertex placed incorrectly, as Since each vertex in A has degree at most $\epsilon 4n$, each contributes at most $\epsilon 4n$ violating edges.

For vertices with high degree, they should be very unlikely to have none of their neighbors in U, so we expect the size of B to be very small. For any vertex with degree at least $\frac{\epsilon}{4}n$, this probability is at most $(1-\epsilon/4)^{|U|}$, so plugging in our choice of $|U|$ this is at most $\epsilon/32$. Consequently, by linearity of expectation, the expected number of vertices in B is at most $\epsilon n/32$. Then by Markov, with probability at least $7/8$, the size of B is at most $\epsilon n/4$.

Adding the contributions from A and B, the expected number of violating edges is at most $\epsilon n^2/2$. Consequently, a quick Chernoff bound on the number of samples shows that the fraction of violating edges we get should be very unlikely to be greater than $3/4$, and so the algorithm will `PASS` with high probability.