Sublinear Time and Space Algorithms 2024A – Lecture 9
Sublinear-Time Algorithms for Sparse Graphs*

Robert Krauthgamer

1 Approximating Average Degree in a Graph

Problem definition:

Input: An n-vertex graph represented (say) as the adjacency list for each vertex (or even just the degree of each vertex).

Goal: Compute the average degree (equiv. number of edges).

Concern: Seems to be impossible e.g. if all degrees ≤ 1, except possibly for a few vertices whose degree is about n.

Theorem 1 [Feige, 2004]: There is an algorithm that estimates the average degree d of a *connected* graph within factor $2+\varepsilon$ in time $O((\frac{1}{\varepsilon})^{O(1)}\sqrt{n/d_0})$, given a lower bound $d_0 \leq d$ and $\varepsilon \in (0, 1/2)$.

We will prove the case $d_0 = 1$ (i.e., it suffices to know G is connected).

Main idea: Use the fact that it is a graph (and not just a list of degrees), although this will show up only in the analysis. A good example to keep in mind is a star graph vs a cycle graph (both have $d \approx 2$).

Algorithm:

1. Choose $s = c\sqrt{n}/\varepsilon^{O(1)}$ vertices at random with replacement, denote this multiset by S and compute the average degree d_S of these vertices.

2. Repeat the above $t = 8/\varepsilon$ times, denoted $S_1, \ldots, S_t$ and report the *smallest* seen estimate $\min_{i\in[t]} d_{S_i}$.

Analysis: We will need 2 lemmas.

Lemma 1a: In each iteration, $\Pr[d_S < (\frac{1}{2} - \varepsilon)d] \leq \varepsilon/64$.

Lemma 1b: In each iteration, $\Pr[d_S > (1 + \varepsilon)d] \leq 1 - \varepsilon/2$.

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

Proof of theorem: Follows easily from the two lemmas, as seen in class.

Proof of Lemma 1b: Follows from Markov's inequality, as seen in class.

Proof of Lemma 1a: Was seen in class, using the fact the degrees form a graph, by considering the high-degree vertices $H \subset V$ and the rest $L = V \setminus H$, and counting edges inside/between them. We saw that a suitable $s = \tilde{O}(\varepsilon^{-2} \max\{|H|, n/|H|\})$ works.

Exer: Explain how to extend the result to any $d_0 \geq 1$.

2 Maximum Matching

Problem definition:

Input: An n-vertex graph $G = (V, E)$ of maximum degree D, represented such that one has direct access to the j-th neighbor of a vertex.

Definition: A matching is a set of edges that are incident to distinct vertices.

Goal: Compute the maximum size of a matching in G.

Note: The matching itself is too large to report in sublinear time, we only estimate its cost using (α, β)-approximation, i.e., $OPT \leq ALG \leq \alpha\ OPT + \beta$.

Theorem 2 [Nguyen and Onak, 2008]: There is an algorithm that gives $(2, \varepsilon n)$ approximation to the maximum matching size in time $D^{O(D)}/\varepsilon^2$.

Main idea: It is well-known that *maximal matching* (note: maximal means with respect to containment) is a 2-approximation for *maximum matching*. We will pick one such matching almost implicitly, and then estimate its size by sampling.

Algorithm GreedyMatching:

1. Start with an empty matching M.

2. Scan the edges (in arbitrary order), and add each edge to M unless it is adjacent to an edge already in M.

Lemma 2a: The output M is a maximal matching, and thus its size is at least half that of a maximum matching.

Exer: Prove this lemma.

Algorithm ApproxGreedyMatching:

1. choose random edge priorities $p(e) \in [0, 1]$, implicitly defining a permutation π of the edges

2. choose $s = O(D/\varepsilon^2)$ edges $e_1, \ldots, e_s$ uniformly at random from the Dn possibilities (note that each edge has two "chances" to be chosen, and some choices may lead to no edge, if the actual degree is smaller than D)

3. for each edge e_i, compute an indicator X_i for whether e_i belongs to the maximal matching M

corresponding to π, by exploring the neighborhood of e_i incrementally

[stop if the algorithm took too many steps altogether]

4. report $X = \frac{Dn}{2s} \sum_i X_i$

Running time: Let M be a greedy matching constructed according to the priorities p (i.e., permutation π). As seen in class, to determine whether a single $e_i \in M$, we only "expect" to explore paths of length up to $k = O(D)$. Thus, the expected running time is $O(sD \cdot D^{cD}) \leq D^{O(D)}/\varepsilon^2$, and by Markov's inequality there is small probability to exceed it by much.

Correctness: As sketched in class, it follows by conditioning on the priorities p (hence the permutation π and matching M are fixed), and applying Chebychev's inequality to $X = \frac{Dn}{2s} \sum_i X_i$ (using the randomness of the s samples).