CS 514: Advanced Algorithms II – Sublinear Algorithms **Rutgers: Spring 2020**

Lecture 2

January 28, 2020

Instructor: Sepehr Assadi *Scribes: Chaitanya Sai Krishna and Vihan Shah*

1 Sublinear Time Algorithms for Graphs

We are going to study sublinear time algorithms in this and the next couple of lectures. In this lecture, we will focus on *graph* algorithms.

1.1 Query Model for Graph Problems

When designing sublinear time algorithms, specifying the exact data model, or rather the *query model*, is very important as the algorithm cannot even read the entire input once[1]. A query model then specifies what type of queries can be made to the input (often times, we assume a query takes $O(1)$ time).

In the context of graph problems, we typically work with one of the following models: *adjacency list* model, *adjacency matrix* model, or the *general query* model. In each model, we assume that the graph $G(V, E)$ has *known* vertices $V = \{1, \ldots, n\}$ (so $\mathrm{ID}(v) \in \{1, \ldots, n\}$ for any $v \in V$) but the edges are unknown. Each model then specifies how one can access the edges of the graph.

Adjacency list query model: The following queries can be answered in $O(1)$ time in this model:

- **Degree queries**: Given a vertex $v \in V$, output $\deg(v)$, namely, the degree of v.
- **Neighbor queries:** Given a vertex $v \in V$ and $i \in [n]$, output the i-th neighbor of v or $\perp$ if $i > \deg(v)$.

By storing the graph in the adjacency list format, we can implement the above query model for algorithms.

Adjacency matrix query model: The following queries can be answered in $O(1)$ time in this model:

- **Pair queries:** Given two vertices $u, v \in V$, output whether (u, v) is an edge in G or not.

By storing the graph in the adjacency matrix format, we can implement the above query model for algorithms.

General query model for graphs: This model is simply a combination of both models above that allows all the three queries mentioned above. This query model can be implemented by storing both the adjacency list and the adjacency matrix of the graph separately.

Remark. The three models above are the most standard models for graph problems. However, sometimes one can consider extensions of these models, for instance, by allowing an extra *edge-sample* query that returns an edge uniformly at random from the graph.

[1]In the classical setting also specifying the input access is important; however, one can typically change different types of access in time *linear* in the input size and so this does not form a barrier for classical algorithms.

Remark. The query models we discussed so far are considered *local* queries as they answer "local" information about the graph (typically functions of only local neighborhood of a single vertex in the graph). Researchers have also studied *global* query models for graph problems that answer much more global information: for instance, given a set S of vertices, return the number of edges with both endpoints among these vertices. We will talk about global queries later in the course and for now only mention that power of local and global queries are vastly different; in particular, there are various problems that can be solved much faster when one has access to these global queries.

2 Estimating Number of Connected Components

We start with one of the most classical problems in the area of sublinear time graph algorithms, namely, estimating the number of connected components, studied first by Chazelle, Rubinfeld, and Trevisan [1]. The problem is as follows:

Problem 1 (Estimating number of connected components). Given a graph $G(V, E)$ in the adjacency list query model, approximation parameter $\varepsilon \in (0, 1)$, and confidence parameter $\delta \in (0, 1)$, output an approximate number of connected components $\widetilde{C}$ such that:

$$\Pr\left(|\widetilde{C} - C| \leq \varepsilon n\right) \geq 1 - \delta,$$

where C is actual number of connected components in G.

Remark. The reason why we settled for this type of approximation, namely, additive approximation (with respect to n) as opposed to multiplicative approximation (having $|\widetilde{C} - C| \leq \varepsilon \cdot C$ instead) or just aiming for the exact answer is as follows. Basically, one can prove that distinguishing whether a graph is connected or has two connected components requires $\Omega(n^2)$ time even with randomization (we will prove this result later in the course). This suggests that we cannot hope to settle even for multiplicative approximation (as any better than two approximation algorithm can distinguish these two cases).

A similar reason also holds for allowing randomization: deterministically, we need $\Omega(n)$ time even for additive approximation algorithms.

Before we get to describe the algorithm, we need some definitions.

Definition 1. For any vertex $v \in V$, we define s_v as the size of the connected component of v in G, i.e., the number of vertices (including v) that are in the same connected component as v.

The following claim reduces the task of estimating the number of connected components to computing a simple function of s_v's for all $v \in V$.

Claim 2. $C = \sum_{v \in V} {}^{1}/_{s_v}$.

Proof. Let $D_1, \ldots, D_C$ denote the connected components of G. Note that $V = D_1 \cup \ldots \cup D_C$ and D_i's are disjoint. Thus,

$$\sum_{v \in V} \frac{1}{s_v} = \sum_{i=1}^{C} \sum_{v \in D_i} \frac{1}{s_v} = \sum_{i=1}^{C} \sum_{v \in D_i} \frac{1}{|D_i|} = \sum_{i=1}^{C} |D_i| \cdot \frac{1}{D_i} = \sum_{i=1}^{C} 1 = C.$$

□

Our general strategy is now to calculate the sum in Claim 2 to estimate C by sampling a small number of vertices v and computing s_v, which can be done by doing a DFS/BFS from v and counting number of visited vertices. This strategy at this point however is problematic because when s_v is very large, computing all vertices connected to v can take a long time. An important observation however is that having a large s_v means that contribution of this vertex to the summation above, i.e., $\frac{1}{s_v}$, is small and thus almost negligible. We formalize this in the following.

Claim 3. *Define $s'_v := \min(s_v, {}^2\!/\!{}_\varepsilon)$ for all $v \in V'$ and $C' := \sum_{v \in V} {}^1\!/\!{}_{s'_v}$. Then, $|C - C'| \le {}^\varepsilon\!/\!{}_2 \cdot n$.*

Proof. First, observe that for each $v \in V$:

$$0 \le \frac{1}{s'_v} - \frac{1}{s_v} \le \frac{\varepsilon}{2}.$$

This holds because $s'_v \le s_v$, $s_v > 0$, and whenever $s'_v \neq s_v$, we have that $s'_v = {}^2\!/\!{}_\varepsilon$, and $s_v > 0$. By summing the inequality over all vertices:

$$C' - C = \sum_{v \in V} \frac{1}{S'_u} - \frac{1}{S_u} \le \frac{\varepsilon}{2} \cdot n,$$
$$C' - C = \sum_{v \in V} \frac{1}{S'_u} - \frac{1}{S_u} \ge 0$$

concluding the proof. □

Claim 3 ensures that if instead of computing s_v, we compute s'_v, we can still get a good estimate of C. However, computing s'_v is easier now since we only need to do a DFS/BFS starting from the vertex v and terminate the search whenever more than ${}^2\!/\!{}_\varepsilon$ vertices are found.

Remark. We should point that Claim 3 gives a straightforward *deterministic* algorithm for this problem – simply compute s'_v for every vertex which takes $O(1/\varepsilon^2)$ per vertex (see Section 2.2 for details). This gives an $O(n/\varepsilon^2)$ time deterministic algorithm which is sublinear in the size of input (which can be $\Omega(n^2)$) but not sublinear in the number of vertices. However, we are going to show that using randomization, one can get a much faster algorithm for this problem.

We are now ready to present the algorithm.

Algorithm:

1. Let $k := {}^2\!/\!{}_{\varepsilon^2} \cdot \ln(2/\delta)$.
2. For $i = 1$ to k do the following:
   - Sample a vertex v_i uniformly at random from v (with replacement).
   - For this v_i compute $X_i := \frac{1}{s'_{v_i}}$ by doing a DFS/BFS from v_i and truncating the search once ${}^2\!/\!{}_\varepsilon$ vertices are visited.
3. Output $\widetilde{C} = {}^n\!/\!{}_k \cdot \sum_{i=1}^{k} X_i$.

In order to analyze this algorithm, we use the following additive variant of Chernoff bound[2]

[2]It is worth mentioning that the bounds one get from multiplicative Chernoff bound is always at least as good as the additive version – we thus only use additive Chernoff for simplifying the calculations when possible.

Proposition 4 (**Additive Chernoff Bound**). *Let $Y_1, Y_2, \ldots, Y_k$ be k independent random variables with values in $[0,1]$ and $Y = \sum_i Y_i$. Then, for any $b \geq 1$,*

$$\Pr[|Y - \mathbb{E}[Y]| > b] \leq 2 \cdot \exp\left(-\frac{2b^2}{k}\right).$$

We now present the proof of correctness and runtime analysis of this algorithm.

2.1 Proof of Correctness

As in the previous lecture, we first compute the expected value of the output, namely, $\widetilde{C}$, and show that it is close to the desired answer and then bound the probability that this random variable deviates significantly from its expectation.

Claim 5. $\mathbb{E}\left[\widetilde{C}\right] = C'$.

Proof. By linearity of expectation, we have,

$$\begin{aligned}\mathbb{E}\left[\widetilde{C}\right] &= \frac{n}{k} \cdot \sum_{i=1}^{k} \mathbb{E}[X_i] = \frac{n}{k} \cdot k \cdot \mathbb{E}[X_1] && \text{(as } X_1, \ldots, X_k \text{ are identically distributed)} \\ &= n \cdot \mathbb{E}[X_1].\end{aligned}$$

We can compute $\mathbb{E}[X_1]$ as follows:

$$\mathbb{E}[X_1] = \sum_{v \in V} \Pr(v \text{ is chosen as } v_1) \cdot \mathbb{E}[X_1 \mid v \text{ is chosen as } v_1] = \frac{1}{n} \cdot \sum_{v \in V} \frac{1}{s'_v} = \frac{1}{n} \cdot C',$$

where the second to last equality is because when we choose v in the algorithm as v_1, we set $X_1 = {}^{1}/_{s'_v}$, and the last equality is by the definition in Claim 3. The claim now follows from the above two equations. ☐

By Claim 5 (and Claim 3), the output is within the desired range in expectation. We now use Chernoff bound to bound the probability that it also deviates from its expectation by much.

Claim 6. $\Pr\left(|\widetilde{C} - C'| \leq \frac{\varepsilon}{2} \cdot n\right) \geq 1 - \delta$.

Proof. Let us define $X = \sum_{i=1}^{k} X_i$. Note that this way $\widetilde{C} = \frac{n}{k} \cdot X$ and $\mathbb{E}[X] = \frac{k}{n} \cdot \mathbb{E}\left[\widetilde{C}\right] = \frac{k}{n} \cdot C'$ by Claim 5. Moreover,

$$|\widetilde{C} - C'| \geq \frac{\varepsilon}{2} \cdot n \iff |\frac{n}{k} \cdot X - \frac{n}{k} \cdot \mathbb{E}[X]| \geq \frac{\varepsilon}{2} \cdot n \iff |X - \mathbb{E}[X]| \geq \frac{\varepsilon}{2} \cdot k.$$

Finally, X is a sum of k independent random variables X_i's which are in $[0,1]$. Hence, we can apply the additive Chernoff bound in Proposition 4 with parameter $b = {}^{\varepsilon}/_{2} \cdot k$ and obtain that,

$$\begin{aligned}\Pr\left(|X - \mathbb{E}[X]| \geq \frac{\varepsilon}{2} \cdot k\right) &\leq 2 \cdot \exp\left(-\frac{2 \cdot (\varepsilon/2)^2 \cdot k^2}{k}\right) = 2 \cdot \exp\left(-\frac{\varepsilon^2}{2} \cdot k\right) \\ &= 2 \cdot \exp\left(-\frac{\varepsilon^2}{2} \cdot \frac{2}{\varepsilon^2} \cdot \ln(2/\delta)\right) \leq 2 \cdot \delta/2 = \delta. && \text{(by the choice of } k)\end{aligned}$$

This proves the desired claim. ☐

By Claim 3 we know that C' is close to C (deterministically) and by Claim 6, we got that $\widetilde{C}$ is close to C' with probability $1 - \delta$. We can combine these two together to conclude the correctness of the algorithm in the following lemma.

Lemma 7. *The output $\widetilde{C}$ of the algorithm satisfies* $\Pr\left(|\widetilde{C} - C| \leq \varepsilon \cdot n\right) \geq 1 - \delta$.

Proof. By Claim 6, with probability $1-\delta$, we have, $|\widetilde{C} - C'| \leq \frac{\varepsilon}{2} \cdot n$. By Claim 3, we have $|C' - C| \leq \frac{\varepsilon}{2} \cdot n$ (deterministically). Hence, by triangle inequality, we have that with probability $1-\delta$,

$$|\widetilde{C} - C| \leq |\widetilde{C} - C'| + |C' - C| \leq \frac{\varepsilon}{2} \cdot n + \frac{\varepsilon}{2} \cdot n = \varepsilon \cdot n,$$

finalizing the proof of correctness of the algorithm. □

2.2 Runtime Analysis

Given v_i, computing s'_{v_i} in the algorithm takes $O(1/\varepsilon^2)$ time because we are going to visit only $2/\varepsilon$ vertices from v_i and thus BFS/DFS will time proportional to number of these vertices plus all edges between them which is at most $O(1/\varepsilon^2)$. As such, the total runtime of the algorithm is:

$$k \cdot O(\frac{1}{\varepsilon^2}) = \frac{1}{\varepsilon^2} \cdot \ln(2/\delta) \cdot O(\frac{1}{\varepsilon^2}) = O(\frac{1}{\varepsilon^4} \cdot \ln(1/\delta)).$$

Remark. Notice that this algorithm runs in constant time (independent of the size of the input graph) whenever ε and δ are fixed constants.

2.3 Concluding Remarks

We saw an algorithm for estimating the number of connected components to within an εn additive approximation in time $O(\frac{1}{\varepsilon^4} \cdot \ln(1/\delta))$. This result was first proved by Chazelle, Rubinfeld, and Trevisan in [1] who used it as a subroutine to estimate the weight of a minimum spanning tree in a graph in sublinear time. We will visit the MST problem later in the course.

Open question? The algorithm we discussed does not seem to obtain optimal bounds as a function of ε, δ. It would be interesting to investigate if these bounds can be improved further and/or prove a matching lower bound for this problem[3].

3 Estimating Average Degree

We now switch to another classical problem defined as follows.

Problem 2 (Estimating average degree). Given a graph G in the adjacency list query model, approximation parameter $\varepsilon \in (0,1)$, and confidence parameter $\delta \in (0,1)$, output an approximate average degree $\widetilde{d}$ such that the following holds:

$$\Pr\left(|\widetilde{d} - \bar{d}| \leq \epsilon \bar{d}\right) \geq 1 - \delta,$$

where $\bar{d}$ is the average degree of G.

Assumption: In this problem, we are going to assume that $\bar{d} \geq 1$[4].

It is worth mentioning that the problem of estimating average degree is equivalent to estimating the number of edges in the graph (since $\bar{d} = \frac{2m}{n}$ and n is given).

[3]Important Note: This problem may have already been solved and a literature search is the first step.
[4]This assumption is needed to obtain a sublinear time algorithm with multiplicative approximation – consider distinguishing a graph with no edges from a one with only a single edge.

This problem was first studied by Feige [2] who gave a $(2+\varepsilon)$-approximation sublinear time algorithm using only degree queries and proved that using only degree queries one cannot obtain a sublinear time algorithm with better than 2-approximation. Subsequently, Goldreich and Ron [3] gave a $(1+\varepsilon)$-approximation algorithm for this problem that also used neighbor queries (i.e., in the adjacency list model). A simpler proof of this result was given by Seshadhri more recently [4]. We will follow the approach of [4] in this lecture note albeit using a different proof.

3.1 Warm Up: Almost-Regular Graphs

As a warm up, let us consider an easy case where the graph is *almost regular*, namely, all vertices have their degree in the interval $[d, 10d]$ for some d known to the algorithm. In general, if we pick a vertex at random and let X be the degree of this vertex, then the expected value of X is $\bar{d}$. This is however not enough to estimate $\bar{d}$ as X can deviate significantly from its expectation. However, in this particular case of almost-regular graphs, we can simply repeat this process multiple times and take the average answer.

Algorithm:

1. Let $k = \frac{50}{\varepsilon^2} \cdot \ln(2/\delta)$.
2. For $i = 1$ to k do the following:
   - Sample a vertex v_i uniformly at random (with replacement)
   - Let $X_i = \deg(v_i)$.
3. Output $\widetilde{d} = \frac{1}{k} \cdot \sum_{i=1}^{k} X_i$.

The runtime of this algorithm is $O(k) = O(1/\varepsilon^2 \cdot \ln(1/\delta))$ since sampling and computing degree of each v_i can be done in $O(1)$ time. We now prove the correctness of the algorithm.

Claim 8. $\mathbb{E}\left[\widetilde{d}\right] = \bar{d}$.

Proof. By linearity of expectation,

$$\mathbb{E}\left[\widetilde{d}\right] = \frac{1}{k} \cdot \sum_{i=1}^{k} \mathbb{E}[X_i] = \mathbb{E}[X_1] \qquad \text{(as } X_1, \ldots, X_k \text{ are identically distributed)}$$
$$= \sum_{v \in V} \frac{1}{n} \cdot \deg(v) = \bar{d}.$$

□

$\widetilde{d}$ in expectation is $\bar{d}$ (which is what we want). We now need to show that $\widetilde{d}$ does not deviate from its expectation by much.

Claim 9. $\Pr(|\widetilde{d} - \bar{d}| \leq \varepsilon \bar{d}) \geq 1 - \delta$

Proof. Note that $\widetilde{d}$ is a sum of independent random variables $X_1, \ldots, X_k$; however, we cannot readily apply Chernoff bound since X_i's are not in $[0,1]$. Instead, we define $Z_i := X_i/10d$ and $Z = \sum_{i=1}^{k} Z_i$, and thus $\widetilde{d} = 10d/k \cdot Z$. Hence, by Claim 8, $\mathbb{E}[Z] = \frac{k}{10d} \cdot \bar{d}$. Note that,

$$|\widetilde{d} - \bar{d}| \geq \varepsilon \bar{d} \iff |\frac{10d}{k} \cdot Z - \frac{10d}{k}\mathbb{E}[Z]| \geq \varepsilon \bar{d} \iff |Z - \mathbb{E}[Z]| \geq \frac{k}{10d} \cdot \varepsilon \bar{d}.$$

Z is a sum of independent random variables in $[0,1]$ (since $\deg(v) \leq 10 \cdot d$ for all $v \in V$ by our simplifying assumption in the warm up). We apply the Chernoff bound in Proposition 4 with parameter $b = \frac{k}{10d} \cdot \varepsilon \bar{d}$:

$$\begin{aligned}
\Pr\left(|Z - \mathbb{E}[Z]| \geq \frac{k}{10d} \cdot \varepsilon \bar{d}\right) &\leq 2 \cdot \exp\left(-\frac{2 \cdot k^2 \cdot \varepsilon^2 \cdot \bar{d}^2}{100 \cdot d^2 \cdot k}\right) \\
&\leq 2 \cdot \exp\left(-\frac{k \cdot \varepsilon^2}{50}\right) \qquad \text{(since } \bar{d} \geq d \text{ by the simplifying assumption)} \\
&= 2 \cdot \exp\left(-\frac{(50/\varepsilon^2) \cdot \ln(2/\delta) \cdot \varepsilon^2}{50}\right) = \delta.
\end{aligned}$$

The above two equations conclude the proof. □

This gives a simple algorithm for the case of almost-regular graphs that takes only $O(1/\varepsilon^2 \cdot \ln(1/\delta))$ time.

3.2 General Case

We now switch to the general case of the problem. What made our algorithm and analysis really easy in the case of almost-regular graphs was that no vertex could contribute significantly to the value of $\bar{d}$ as degrees of all vertices were close to each other. In the general case however, we need to take care of vertices that have much higher degree than the remaining ones; if you repeat the above algorithm in this case, the best bound that follows is $O(n/\varepsilon^2 \cdot \ln(1/\delta))$ (as in any graph the degrees are within at most factor n of each other). But this is completely trivial as in $O(n)$ time we can simply query degree of all vertices!

In the following, we are going to use the fact that a graph with m edges cannot have "too many" vertices with "too large" degree, and use this to reduce the "noise" introduced by high degree vertices. For this purpose, we will assign the edges to their lower-degree endpoint (breaking the ties arbitrarily but consistently) and show how to use this to reduce the variance of our estimator. We first need some definition.

Definition 10. We define a total ordering $\prec$ on vertices of G where $u \prec v$ if and only if either $\deg(u) < \deg(v)$ or $\deg(u) = \deg(v)$ and $\mathrm{ID}(u) < \mathrm{ID}(v)$ (just to break the ties consistently – here $\mathrm{ID}(u)$ refers to the label of u in $\{1, \ldots, n\}$). Moreover, for any $u \in V$, we define $\deg^+(u)$ as the number of neighbors v of u where $u \prec v$.

We can now present the algorithm.

Algorithm:

```
1. Let $k = \frac{16}{\varepsilon^2} \cdot \sqrt{n}$.
2. For $i = 1$ to $k$ do the following:
     • Sample a vertex $v_i$ uniformly at random from the graph.
     • Sample a vertex $u_i \in N(v)$ uniformly at random from the neighbors of $v$.
     • If $v_i \prec u_i$ then $X_i = 2 \cdot \deg(v_i)$ else $X_i = 0$.
3. return $\widetilde{d} = \frac{1}{k} \cdot \sum_{i=1}^{k} X_i$.
```

To emphasize the intuition behind the algorithm again, we are sampling an edge (u_i, v_i) from the graph G, although *not* uniformly at random, and count this edge if it is assigned to v_i (and scale the random variable by $2 \cdot \deg(v_i)$), and otherwise ignore this edge.

We now formalize this intuition.

Remark. The proof of this result follows a general framework. Design a random variable X (think of each X_i in the above) where $\mathbb{E}[X]$ is equal to the desired output and $\text{Var}[X]$ is not "too high". Then repeat this process independently sufficiently many times (based on the bound on the variance) to get random variables $X_1, \ldots, X_k$ and return Y = average of these random variables as the answer. This averaging step keeps the expected value of Y the same as expected value of X and hence the same as the desired answer but reduces the variance of Y (compared to X) by a factor of k. As such, we can now apply Chebyshev's inequality and bound the probability of deviation of Y from its expectation (we need to pick the number of random variables to average, i.e., k, large enough to adjust the probability bound we get from the Chebyshev's inequality).

We follow the approach outlined in the remark above. Let X denote the random variable corresponding to value of each X_i in the for-loop of the algorithm (note that X_i's are identically distributed). We bound expectation and variance of X in the following.

Claim 11. $\mathbb{E}[X] = \bar{d}$.

Proof. We have,

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{v \in V} \Pr(v \text{ is sampled from } V) \cdot \mathbb{E}[X_1 \mid v \text{ is sampled from } V] = \sum_{v \in V} \frac{1}{n} \cdot \mathbb{E}[X_1 \mid v \text{ is sampled from } V] \\
&= \frac{1}{n} \sum_{v \in V} \sum_{u \in N(v)} \Pr(u \text{ is sampled in } N(v) \mid v \text{ is sampled}) \, \mathbb{E}[X_1 \mid u \text{ is sampled in } N(v) \text{ and } v \text{ is sampled}] \\
&= \frac{1}{n} \sum_{v \in V} \sum_{u \in N(v) \wedge v \prec u} \frac{1}{\deg(v)} \cdot 2\deg(v) \qquad (X_1 = 2\deg(v) \text{ whenever } v \prec u \text{ and otherwise is zero}) \\
&= \frac{1}{n} \sum_{v \in V} 2 \cdot \deg^+(v) = \frac{2m}{n} = \bar{d}, \qquad (\text{by definition, } \deg^+(v) \text{ counts } u \in N(v) \text{ with } v \prec u)
\end{aligned}$$

where we used the fact that $\sum_{v \in V} \deg^+(v) = m$ as every edge is counted exactly once (by its lower rank endpoint) in this sum. □

Lemma 12. $\text{Var}[X] \leq 4\sqrt{2m} \cdot \bar{d}$.

Before we prove Lemma 12, we need two intermediate claims. Define $H \subseteq V$ as the set of first $\sqrt{2m}$ vertices with largest rank according to the ordering $\prec$ (these are the highest degree vertices of G). Also, define $L := V \setminus H$ to be the remaining vertices. We bound $\deg^+$ of vertices in H and deg of vertices in L as follows.

Claim 13. *For any $v \in H$, $\deg^+(v) \leq \sqrt{2m}$.*

Proof. By definition, $\deg^+(v)$ counts the neighbors of v with rank higher than v. Since v itself is among the first $\sqrt{2m}$ highest rank vertices, $\deg^+(v) \leq \sqrt{2m}$. □

Claim 14. *For any $v \in L$, $\deg(v) \leq \sqrt{2m}$.*

Proof. Suppose by contradiction that there is a vertex $v \in L$ such that $\deg(v) > \sqrt{2m}$. This implies that *all* vertices in H have degree at least $\sqrt{2m}$ since degrees of every vertex in H is at least as large as any vertex in L. But by simply summing up the degrees of vertices in H we obtain that the graph needs to have $> \frac{1}{2} \cdot \sqrt{2m} \cdot \sqrt{2m} > m$ edges, a contradiction with m being the number of edges. □

We are now ready to bound the variance of X and prove Lemma 12.

Proof of Lemma 12. By definition of variance, we have,

$$\mathrm{Var}\left[X\right] = \mathbb{E}\left[X^2\right] - \mathbb{E}\left[X\right]^2 \leq \mathbb{E}\left[X^2\right] = \frac{1}{n}\sum_{v \in V}\sum_{u \in N(v) \wedge v \prec u} \frac{1}{\deg(v)} \cdot (2\deg(v))^2$$

(the calculation is exactly the same as expectation in Claim 11 by considering $X^2 = (2\deg(v))^2$ instead)

$$= \frac{4}{n}\sum_{v \in V} \deg^+(v) \cdot \deg(v) \qquad \text{(by definition of } \deg^+(v))$$

$$= \frac{4}{n}\left(\sum_{v \in H} \deg^+(v) \cdot \deg(v) + \sum_{v \in L} \deg^+(v) \cdot \deg(v)\right)$$

$$= \frac{4}{n}\left(\sum_{v \in H} \sqrt{2m} \cdot \deg(v) + \sum_{v \in L} \deg^+(v) \cdot \sqrt{2m}\right) \qquad \text{(by Claims 13 and 14)}$$

$$\leq \frac{4\sqrt{2m}}{n} \cdot \left(\sum_{v \in H} \deg(v) + \sum_{v \in L} \deg(v)\right) = 4 \cdot \sqrt{2m} \cdot \bar{d}.$$

$$\text{(as } \deg^+(v) \leq \deg(v) \text{ and } \textstyle\sum_{v \in V} \deg(v) = 2m)$$

□

We can now finalize the proof of correctness of the algorithm. For that, we need the following simple claim about variance (alluded to already in the remark about the general approach).

Proposition 15. *Let $Y = \frac{1}{k}\sum_{i=1}^{k} X_i$ be an average of k independent copies of a random variable X. Then,*

$$\mathrm{Var}\left[Y\right] = \frac{1}{k} \cdot \mathrm{Var}\left[X\right].$$

Proof. We have,

$$\mathrm{Var}\left[Y\right] = \mathrm{Var}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right] = \frac{1}{k^2} \cdot \mathrm{Var}\left[\sum_{i=1}^{k} X_i\right] \qquad (\mathrm{Var}\left[c \cdot X\right] = c^2 \cdot X \text{ for any scalar } c)$$

$$= \frac{1}{k^2} \cdot \sum_{i=1}^{k} \mathrm{Var}\left[X_i\right] \qquad (\mathrm{Var}\left[A+B\right] = \mathrm{Var}\left[A\right] + \mathrm{Var}\left[B\right] \text{ for independent random variables } A, B)$$

$$= \frac{1}{k^2} \cdot k \cdot \mathrm{Var}\left[X\right] = \frac{1}{k} \cdot \mathrm{Var}\left[X\right].$$

□

We can now finalize the proof of correctness.

Lemma 16. *For the output $\widetilde{d}$ of the algorithm,* $\Pr\left(|\widetilde{d} - \bar{d}| \leq \varepsilon \cdot \bar{d}\right) \geq 3/4$.

Proof. By Claim 11 and linearity of expectation $\mathbb{E}\left[\widetilde{d}\right] = \bar{d}$. By Lemma 12 and Proposition 15, we further have $\mathrm{Var}\left[\widetilde{d}\right] \leq \frac{4\sqrt{2m}}{k} \cdot \bar{d}$. By Chebyshev's inequality,

$$\Pr\left(|\widetilde{d} - \bar{d}| \geq \varepsilon \cdot \bar{d}\right) = \Pr\left(|\widetilde{d} - \mathbb{E}\left[\widetilde{d}\right] | \geq \varepsilon \cdot \bar{d}\right) \leq \frac{\mathrm{Var}\left[\widetilde{d}\right]}{\varepsilon^2 \cdot \bar{d}^2} \leq \frac{4\sqrt{2m} \cdot \bar{d}}{k \cdot \varepsilon^2 \cdot \bar{d}^2}$$

$$= \frac{4\sqrt{2m} \cdot n}{k \cdot \varepsilon^2 \cdot 2m} = \frac{4 \cdot n}{k \cdot \varepsilon^2 \cdot \sqrt{2m}} = \frac{\sqrt{n}}{4 \cdot \sqrt{2m}} \qquad \text{(by the choice of } k = \tfrac{16}{\varepsilon^2} \cdot \sqrt{n})$$

$$\leq \frac{1}{4}. \qquad \text{(by the assumption that } \bar{d} = 2m/n \geq 1)$$

□

Lemma 16 implies that the algorithm above gives a $(1 \pm \varepsilon)$-approximation to the average degree in $O(\frac{\sqrt{n}}{\varepsilon^2})$ time with probability 3/4.

3.3 Amplifying the Probability of Success

So far, we obtained an algorithm for solving the problem with constant probability (specifically 3/4) in $O(\frac{\sqrt{n}}{\varepsilon^2})$ time. However, recall that our goal was to have a confidence probability of $1 - \delta$.

One ad-hoc way of fixing this is to change the choice of k in the algorithm to $k = \frac{16}{\varepsilon^2} \cdot \sqrt{n} \cdot \frac{1}{\delta}$, namely increase it by a factor of $1/\delta$. The proof then follows by a very basic modification of Lemma 16. Nevertheless, this approach, beside being rather naive and ad-hoc, will also result in increasing the time complexity of the algorithm by $O(1/\delta)$ which is not desirable. Instead, we are going to suggest a general approach that is applicable to almost every problem in a black-box way (and in many other algorithmic problems beside sublinear algorithms). This is often times referred to as the *median trick*.

Median Trick. Consider the algorithm discussed in the previous section. In order to boost or amplify its probability of success, we do the following: (i) Run the algorithm independently for $t = 8 \ln(2/\delta)$ times and record the answer of i-th time as X_i; (ii) Return the median of X_i's as the final answer.

Lemma 17. *The output $\bar{d}$ of the new algorithm satisfies* $\Pr\left(|\widetilde{d} - \bar{d}| \leq \varepsilon \cdot \bar{d}\right) \geq 1 - \delta$.

Proof. For $\bar{d}$, the median answer of X_i's, to be out of the desired range $|\widetilde{d} - \bar{d}| \leq \varepsilon \cdot \bar{d}$, one of the following events should happen:

- Event 1: At least half the X_i's are smaller than $(1 - \varepsilon) \cdot \bar{d}$;
- Event 2: At least half the X_i's are larger than $(1 + \varepsilon) \cdot \bar{d}$.

We bound the probability of each event now.

Define the indicator random variable $Z_i \in \{0, 1\}$ where $Z_i = 1$ if and only if $X_i < (1 - \varepsilon) \cdot \bar{d}$ and define $Z = \sum_{i=1}^{t} Z_i$. By Lemma 16, $\mathbb{E}[Z_i] \leq 1/4$ and thus $\mathbb{E}[Z] \leq t/4$. Since Z is a sum of independent random variables in $[0, 1]$, by Chernoff bound in Proposition 4 (with $b = \frac{t}{4}$),

$$\Pr(\text{Event 1}) = \Pr\left(Z \geq \frac{t}{2}\right) \leq \Pr\left(|Z - \mathbb{E}[Z]| \geq \frac{t}{4}\right) \leq \exp\left(-\frac{2t^2}{16 \cdot t}\right) = \exp(-\ln(2/\delta)) = \frac{\delta}{2}.$$

One can similarly define random variables $W_i \in \{0, 1\}$ where $W_i = 1$ if and only if $X_i > (1 + \varepsilon) \cdot \bar{d}$ and define $W = \sum_{i=1}^{t} W_i$. The same exact argument as above implies that,

$$\Pr(\text{Event 2}) = \Pr\left(W \geq \frac{t}{2}\right) \leq \Pr\left(|W - \mathbb{E}[W]| \geq \frac{t}{4}\right) \leq \frac{\delta}{2}.$$

A union bound on the two events now finalizes the proof. $\square$

Using the approach above, we can solve the degree estimation problem in $O(\frac{\sqrt{n}}{\varepsilon^2} \cdot \ln(1/\delta))$ with probability $1 - \delta$ for any arbitrary $\delta > 0$.

Remark. It is worth emphasizing that the median trick completely treated the algorithm (and analysis) of the main subroutine from the previous section in a black-box way. One can use this technique to boost the probability of success of *any* algorithm (in many different settings) – this is indeed the reason that for most algorithms, the dependence of resources on δ is almost always $O(\ln(1/\delta))$.

References

[1] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005. 2, 5

[2] U. Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM J. Comput.*, 35(4):964–984, 2006. 6

[3] O. Goldreich and D. Ron. Approximating average parameters of graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008. 6

[4] C. Seshadhri. A simpler sublinear algorithm for approximating the triangle count. *CoRR*, abs/1505.01927, 2015. 6