Sublinear-Space Streaming Algorithms for Estimating Graph Parameters on Sparse Graphs

Xiuge Chen

School of Computing and Information Systems, The University of Melbourne, Australia

Rajesh Chitnis

University of Birmingham, UK

Patrick Eades

School of Computing and Information Systems, The University of Melbourne, Australia

Anthony Wirth

School of Computing and Information Systems, The University of Melbourne, Australia

Abstract

Fundamental graph problems have been studied extensively in the RAM model of computation. Over the last decade, there have been great advances in streaming algorithms for such graph problems. In this paper, we design new streaming algorithms for three problems: maximum independent set, minimum dominating set, and maximum matching. Our methods approximate these three graph parameters in sparse graph classes, i.e., where the number of edges is linear in the number of vertices. Since our aim is sublinear working-space algorithms, and each of the three parameters can have linear size even in sparse graphs, we can only estimate these parameters instead of actually producing solutions in our working space. We obtain the following two families of results:

- **Estimating Max Independent Set via the Caro-Wei bound**: Caro and Wei each showed that $\lambda = \sum_v 1/(d(v)+1)$ lower bounds the size of a max independent set, where $d(v)$ is the degree of vertex v. For graphs with constant average degree, $\bar{d}$, and with maximum degree $\Delta = \mathcal{O}(\varepsilon^2 \bar{d}^{-3} n)$, we develop algorithms that, with at least $1-\delta$ success probability:
  - In the *online streaming* model, return an independent set of size $1 \pm \varepsilon$ times λ. This improves on results by Halldórsson et al. [Algorithmica '16], with less space, i.e., $\mathcal{O}(\log \varepsilon^{-1} \cdot \log n \cdot \log \delta^{-1})$, faster update time, i.e., $\mathcal{O}(\log \varepsilon^{-1})$, and bounded success probability.
  - In insertion-only streams, approximate the Caro-Wei bound within a factor of $1 \pm \varepsilon$, in one pass and in $\mathcal{O}(\bar{d}\varepsilon^{-2} \log n \cdot \log \delta^{-1})$ space. This aligns with the state-of-art result of Cormode et al. [ISCO '18], though our method also works in the *online streaming* model. If the stream is vertex-arrival and random-order, the space can be further reduced to $\mathcal{O}(\log(\bar{d}\varepsilon^{-2}))$. Trading off extra space and post processing, we can remove the max-degree constraint.
- **Sublinear-Space Algorithms on Forests**: When the input is a forest, Esfandiari et al. [SODA '15, TALG '18] showed space lower bounds[1] of $\Omega(\sqrt{n})$, for randomized algorithms, for estimating our three graph parameters. We narrow the gap between upper and lower bounds, with algorithms that approximate:
  - The size of a max independent set within $3/2 \cdot (1 \pm \varepsilon)$ in one pass and in $\log^{\mathcal{O}(1)} n$ space, and within $4/3 \cdot (1 \pm \varepsilon)$ in two passes and in $\tilde{\mathcal{O}}(\sqrt{n})$ space; the lower bound of Esfandiari et al. holds for approximation better than $4/3$.
  - The size of a min dominating set within $3 \cdot (1 \pm \varepsilon)$ in one pass and in $\log^{\mathcal{O}(1)} n$ space, and within $2 \cdot (1 \pm \varepsilon)$ in two passes and in $\tilde{\mathcal{O}}(\sqrt{n})$ space; the lower bound of Esfandiari et al. holds for approximation better than $3/2$.
  - The size of a max matching within $2 \cdot (1 \pm \varepsilon)$ in one pass and in $\log^{\mathcal{O}(1)} n$ space, and within $3/2 \cdot (1 \pm \varepsilon)$ in two passes and in $\tilde{\mathcal{O}}(\sqrt{n})$ space; the lower bound of Esfandiari et al. holds for approximation better than $3/2$.

---

[1] In fact, the lower bounds hold even when each component is a path!

2012 ACM Subject Classification Theory of computation → Streaming models; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Streaming Algorithms, Sublinear Space, Graph Problems

1 Introduction

Numerous real-world problems can be nicely modeled and solved as graphs. Consequently, over the last century or so, a wide range of graph problems have been studied, such as independent set, dominating set, and matching. For example, independent set models optimization and scheduling problems where conflicts should be avoided [31, 1, 21, 24], dominating set models guardian selection problems [43, 36, 38, 47, 42], while a matching models similarity [45, 14, 20] and inclusion dependencies [3, 16].

Parameters of interest: Given an undirected graph G, comprising vertices V and edges E, let n and m be the sizes of V and E, respectively. A subset S of V, is an independent set if and only if the subgraph induced by S contains no edges. Alternatively, S is a dominating set if and only if every vertex in V is either in S itself or adjacent to some vertex in S. A subset M of edges E is a matching if and only if no pair of edges in M shares a vertex. The three parameters of interest to us are the size of a maximum independent set, which we denote by the independence number, β; the size of a minimum dominating set, which we denote by the domination number, γ; and finally the size of a maximum matching, which we denote by the matching number, ϕ. It is well known that $n - \phi \geq \beta \geq n - 2\phi$.

Deciding whether each of these parameters is larger (for max) or smaller (for min) than a specified parameter is NP-complete [30]. Aligned with numerous other graph optimization problems, there are many *approximation algorithms* that run in polynomial time and return a solution within a guaranteed factor of optimum. For instance, a greedy algorithm for maximum matching outputs a 2-approximation. Similarly, there exist greedy algorithms that $\mathcal{O}(\Delta)$-approximate the maximum independent set [23] and approximate within $\mathcal{O}(\ln \Delta)$ the minimum dominating set [27, 33], where Δ is the maximum degree. More promising results are obtained on sparse graphs, i.e., graphs with a linear number of edges, which are the objects of study in this paper. It is known, e.g., that the Caro-Wei bound [7, 46], given by $\sum_{v \in V(G)} \frac{1}{1+\deg(v)}$, and which we call λ, is a lower bound on β.

Data streams: We focus on estimating these graph parameters in the data stream model. In the *data stream* model, instead of being pre-stored in memory, data elements arrive sequentially, and the available memory is typically much less than the complete data set size. The *semi-streaming* model [19], where $\mathcal{O}(n \log n)$ bits are allowed, is commonplace for general graphs. In this paper, we consider *sparse* graphs, so we restrict our space allowance to $o(n)$ bits as $\mathcal{O}(n)$ bits is sufficient to store the entire graph. There are multiple well-studied settings for graph streaming, including edge and vertex arrival, arbitrary or random order, and insertion only versus turnstile. Halldórsson et al. [22] introduced the *online streaming* model, combining the data-stream and online models. In particular, we distinguish between the *solution space*, for storing the solution and the *working space* for determining it. Section A.1 includes formal definitions of the *data stream* and *online streaming* models.

1.1 Previous Results

Estimating the size of a maximum matching in a sparse graph has been extensively studied in the streaming setting. When the input is a tree, several $(2+\varepsilon)$-approximation algorithms are known. Esfandiari et al. [18] designed an $\tilde{\mathcal{O}}(\sqrt{n})$-space algorithm for insertion-only streams, where $\tilde{\mathcal{O}}$ notation suppresses a poly-logarithmic factor. The space was further reduced to $\log^{\mathcal{O}(1)} n$ by Cormode et al. [12], while Bury et al. [5] generalized it to turnstile streams.

However, very little research has been done for independent set and dominating set. Halldórsson et al. [22] studied general hypergraphs and gave a one-pass insertion-only streaming algorithm in $\mathcal{O}(n)$ space, outputting an independent set with size at least the Caro-

Wei bound, λ, in expectation. Their algorithm suits the *online streaming* model. Cormode et al. [10] designed an algorithm that $(1 \pm \varepsilon)$-approximates λ with constant success probability in $\mathcal{O}(\varepsilon^{-2}\bar{d}\log n)$ space. They also showed a nearly tight lower bound: every randomized one-pass algorithm with constant error probability requires $\Omega(\varepsilon^{-2}\bar{d})$ space to c-approximate λ. Meanwhile, Esfandiari et al. [18] established space lower bounds for estimating ϕ in a forest. For every randomized streaming approximation algorithm with factor better than $3/2$, we need $\Omega(\sqrt{n})$ space. Their techniques give lower bounds for other parameters: $\Omega(\sqrt{n})$ bits are required to approximate β better than $4/3$, and γ better than $3/2$.

1.2 Our Results

We introduce new algorithms for independent set, dominating set, and matching in streamed sparse graphs, including graphs with bounded average degree (e.g., graphs with bounded arboricity and planar graphs) and forests. Our results fall into two categories.

1.2.1 Approximating the Caro-Wei bound

We approximate the independence number in graphs with bounded average degree via the Caro-Wei bound, λ. As Boppana et al. [4] show, λ is at least the Turán Bound [44], which is the bound $\beta \geq n/(\bar{d}+1)$). Hence, every $\mathcal{O}(1)$-approximation of λ gives a $\mathcal{O}(\bar{d})$-approximation of β. Our results are summarized in Table 1.

For graphs with average degree $\bar{d}$ and max degree $\Delta \in \mathcal{O}(\varepsilon^2\bar{d}^{-3}n)$, we give an arbitrary-order edge-arrival algorithm, Caraway, that $(1\pm\varepsilon)$-approximates λ. Our algorithm fails with probability at most δ, and takes space $\mathcal{O}(\bar{d}\varepsilon^{-2}\log n \cdot \log \delta^{-1})$. More importantly, Caraway can be modified to report an actual solution set in the *online streaming* model with $\mathcal{O}(\log \varepsilon^{-1})$ update time and working space $\mathcal{O}(\log \varepsilon^{-1} \log n \cdot \log \delta^{-1})$. Also, we can remove the max-degree limitation can be removed if only a guarantee in expectation.

Alternatively, with a slightly worse approximation ratio and space requirement, we can remove the max-degree assumption. Failing with low probability, i.e., at most n^{-c} for some constant c, the new algorithm asks for $\mathcal{O}(\bar{d}^4\varepsilon^{-2}\log^2 n)$ space, and returns a $(1-\varepsilon)(1\pm\varepsilon)$-approximation. Since post-processing is a component, this version of Caraway cannot be used in *online streaming* model. However, if the stream is vertex arrival and random-order, then the space need of our algorithm can be further reduced to only $\mathcal{O}(\log(\bar{d}\varepsilon^{-2})\log \delta^{-1})$.

There is a known offline algorithm for estimating λ, based on a random permutation of the vertices. Caraway simulates a random permutation via hash functions drawn uniformly at random from an ε-min-wise hash family[2]. In the analysis, bounding what we call the approximation error *rate* [3], ε, is non-trivial, due to positive correlation between some pairs of vertices. We provide the first analysis that bounds ε, when the max degree is not too large. Our results combine the advantages of Cormode et al. [10] and Halldórsson et al. [22].

Contrast with Cormode et al. [10] : Algorithms by Cormode et al. [10] report β, but do not report an actual independent set. When the stream is insertion only and the maximum degree is not too large, Caraway achieves asymptotically the same approximation ratio in the same space as Cormode et al. [10] do. More importantly, Caraway can be modified to output an independent set solution in the *online streaming* model. Moreover, in the vertex-arrival and random-order models, Caraway requires much less space than the method of Cormode et al. [10].

[2] See Section A.2 for the definition of ε-min-wise hash families.
[3] The relative error between the estimate and the actual value.

Table 1 Streaming Caro-Wei Bound Approximation. Arrivals are *Edge* for edge-arrival or *Vertex* for vertex-arrival. Types are *Insert.* for insertion-only or *Turns.* for turnstile. Orders are *Arb.* for arbitrary or *Ran.* for random. Online is "Yes" if it is an *online streaming* algorithm. Prob. indicates success probability, where c is an arbitrary constant. Theorems marked with '*' require the max vertex degree to be at most $\frac{\varepsilon^2 n}{3(\bar{d}+1)^3}$: if an *in-expectation guarantee* suffices, we remove this condition.

Arrival	Type	Order	Factor	(Work) Space	Online	Prob.	Reference
Edge	Insert.	Arb.	1	$\mathcal{O}(n)$	Yes	*Expected*	[22]
Edge	Turns.	Arb.	$(1 \pm \varepsilon)$	$\mathcal{O}(\bar{d}\varepsilon^{-2} \log n)$	No	c	[10]
Edge	Insert.	Arb.	$(1 \pm \varepsilon)$	$\mathcal{O}(\bar{d}\varepsilon^{-2} \log n)$	No	c	Theorem 5*
Edge	Insert.	Arb.	$(1 \pm \varepsilon)$	$\mathcal{O}(\log \varepsilon^{-1} \log n)$	Yes	c	Theorem 9*
Edge	Insert.	Arb.	$(1-\varepsilon)(1 \pm \varepsilon)$	$\mathcal{O}(\bar{d}^2\varepsilon^{-2} \log^2 n)$	No	n^{-c}	Theorem 12
Vertex	Insert.	Ran.	$(1 \pm \varepsilon)$	$\mathcal{O}(\log(\bar{d}\varepsilon^{-2}))$	No	c	Theorem 13*
Either	Either	Arb.	c	$\Omega(\bar{d}/c^2)$	-	c	[10]

Table 2 Streaming Forests Algorithms. Each algorithm succeeds with high probability.

Problem	Arrival	Type	Order	Pass	Factor	Space	Reference
γ	Edge	Turnstile	Arb.	1	$3(1 \pm \varepsilon)$	$\log^{\mathcal{O}(1)} n$	Theorem 26
γ	Edge	Turnstile	Arb.	2	$2(1 \pm \varepsilon)$	$\tilde{\mathcal{O}}(\sqrt{n})$	Theorem 33
γ	Any	Any	Arb.	1	$3/2 - \varepsilon$	$\Omega(\sqrt{n})$	[18]
β	Edge	Turnstile	Arb.	1	$3/2(1 \pm \varepsilon)$	$\log^{\mathcal{O}(1)} n$	Theorem 29
β	Edge	Turnstile	Arb.	2	$4/3(1 \pm \varepsilon)$	$\tilde{\mathcal{O}}(\sqrt{n})$	Theorem 32
β	Any	Any	Arb.	1	$4/3 - \varepsilon$	$\Omega(\sqrt{n})$	[18]
ϕ	Edge	Insertion	Arb.	1	$2(1 \pm \varepsilon)$	$\tilde{\mathcal{O}}(\sqrt{n})$	[18]
ϕ	Edge	Insertion	Arb.	1	$2(1 \pm \varepsilon)$	$\log^{\mathcal{O}(1)} n$	[12]
ϕ	Edge	Turnstile	Arb.	1	$2(1 \pm \varepsilon)$	$\log^{\mathcal{O}(1)} n$	[6], Theorem 25
ϕ	Edge	Turnstile	Arb.	2	$3/2(1 \pm \varepsilon)$	$\tilde{\mathcal{O}}(\sqrt{n})$	Theorem 34
ϕ	Any	Any	Arb.	1	$3/2 - \varepsilon$	$\Omega(\sqrt{n})$	[18]

Contrast with Halldórsson et al. [22] : The online streaming algorithm by Halldórsson et al. [22] has update time $\mathcal{O}(\log n)$ in working space $\mathcal{O}(n)$. CARAWAY has lower update time, $\mathcal{O}(\log \varepsilon^{-1})$, and less working space, i.e., $\mathcal{O}(\log \varepsilon^{-1} \cdot \log n \cdot \log \delta^{-1})$. Besides, we offer a guaranteed probability that our estimate is within error rate, ε; there is no such guarantee from Halldórsson et al. [22].

1.2.2 Estimating β, γ and ϕ on Forests

Our results on estimating the independence number β, domination number γ and matching number ϕ on forests in the streaming model are summarized in Table 2. We trade off approximation ratio for space and number of passes, and obtain two classes of results:

- **One-pass $\log^{\mathcal{O}(1)} n$ space algorithms**:
  - a $3/2 \cdot (1 \pm \epsilon)$-approximation for the independence number, β
  - a $3 \cdot (1 \pm \epsilon)$-approximation for the domination number, γ
  - a $2 \cdot (1 \pm \epsilon)$-approximation for the matching number, ϕ.

We rely on approximating the numbers of both leaves[4] and non-leaf vertices in a stream.

[4] A leaf is a vertex with degree one.

- **Two-pass $\tilde{\mathcal{O}}(\sqrt{n})$ space**[5]:
  - a $4/3 \cdot (1 \pm \epsilon)$-approximation algorithm for β
  - a $2 \cdot (1 \pm \epsilon)$-approximation algorithm for γ
  - a $3/2 \cdot (1 \pm \epsilon)$-approximation algorithm for ϕ.

Our results further close the gap between the upper and lower bounds of Esfandiari et al. [18]. We rely on the notion of support vertices, which is found in discrete mathematics. A support vertex is one that is adjacent to one or more leaves. We believe that this useful quantity can be applied to design algorithms for many other tree and forest problems.

We also remark that our techniques can be easily adapted to other computational models. For example, in distributed settings, we can achieve the same results as above in constant (one or two) communication rounds with only $\mathcal{O}(1)$ message size.

1.3 Further Related Streaming Work

On graphs with bounded arboricity, α, there are approximation algorithms for estimating ϕ. For insertion-only streams, Esfandiari et al. [18] developed a $(5\alpha+9)$-approximation algorithm that requires $\tilde{\mathcal{O}}(\alpha n^{2/3})$ space. Cormode et al. [12] traded off approximation ratio for space, and showed a $(22.5\alpha+6)$-approximation algorithm in only $\mathcal{O}(\alpha \log^{\mathcal{O}(1)} n)$ space. McGregor and Vorotnikova [34] showed that the algorithm of Cormode et al. [12] can achieve a $(\alpha+2)$-approximation via a tighter analysis. For turnstile streams, Chitnis et al. [9] designed a $(22.5\alpha+6)$-approximation algorithm using $\tilde{\mathcal{O}}(\alpha n^{4/5})$ space. Bury et al. [5] improved the approximation ratio to $(\alpha+2)$, their algorithm asks for $\tilde{\mathcal{O}}(\alpha n^{4/5})$ space in edge-arrival streams, but only $\mathcal{O}(\log n)$ in vertex-arrival streams.

Relaxing the arrival order from arbitrary to random, Monemizadeh et al. [37] converted known constant-time RAM-model approximation algorithms into constant-space streaming algorithms. Their algorithm approximates γ in bounded-degree graphs, with an additive error term of εn. Peng and Sohler [41] further extended this result to graphs of bounded average degree. They also showed that in planar graphs β can be $(1+\varepsilon)$-approximated using constant, but still massive space, i.e., $\mathcal{O}(2^{(1/\varepsilon)^{(1/\varepsilon)^{\log^{\mathcal{O}(1)}(1/\varepsilon)}}})$. Cormode et al. [10] also showed that in a vertex-arrival stream with very large average degree, there exists an algorithm that $(\log n)$-approximates the Caro-Wei Bound using $\mathcal{O}(\log^3 n)$ space.

Meanwhile, Chitnis and Cormode [8] adapted the lower-bound reduction technique from Assadi et al. [2] and showed that, for graphs with arboricity $\alpha+2$, for $\alpha \geq 1$, every randomized $\alpha/32$-approximation algorithm for minimum dominating set requires $\Omega(n)$ space. This lower bound holds even under the vertex-arrival model.

2 Preliminaries

2.1 Graph and Problem Definitions

For an undirected graph, G, given a subset of its vertices, $S \subseteq V(G)$, let $N(S)$ be the neighbors of S (excluding S itself), and $N[S]$ be $S \cup N(S)$. Let $d(v)$ be the degree of vertex v, Δ be the max degree in G, and $\bar{d}$ be the average degree in G. Let $Deg_i(G)$ be the subset of V with $d(v)=i$, and $Deg_{\geq i}(G)$ be the subset of V with $d(v) \geq i$. Many of the

[5] The symbol $\tilde{\mathcal{O}}$ suppresses the poly-logarithmic factor in the bound.

estimates in this paper use the notion of a support vertex [15] which is defined as a vertex that is adjacent to at least one leaf. For a graph G, we denote it by $Supp(G)$.

2.2 Fundamental Streaming Algorithms

To support effective algorithm development, we have several assumptions about the stream. For convenience, consistent with prior art, we assume the vertices are $[n] = \{1, 2, \ldots, n\}$, and that the value of n is known in advance. Second, our results are developed for certain graph classes, hence we assume that at the *end* of the stream, the graph is guaranteed to be in the target class. But there is no such requirement *during* the stream.

We employ several streaming primitives. Consider a vector $x \in \mathbb{R}^n$, whose coordinates are updated by a turnstile stream, where increments can be negative. Consistent with prior art, assume each update is in the range $[-M, M]$. Let x_i be the final value of coordinate i. For $p > 0$, the L_p norm of x is $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$. The L_0 *norm* of x is $\|x\|_0 = (\sum_{i=1}^n |x_i|^0)$. Both L_p norm and L_0 *norm* can be $(1 \pm \varepsilon)$-approximated in streams in small space.

▶ **Theorem 1.** *[28] For all $p \in (0, 2)$, given $\varepsilon, \delta \in (0, 1)$, there exists an algorithm that uses $\mathcal{O}(\varepsilon^{-2} \log(mM) \log \delta^{-1})$ space and with probability at least $1 - \delta$ reports a value in the range $(1 \pm \varepsilon) \|x\|_p$. Both update time and reporting time are in $\tilde{\mathcal{O}}(\varepsilon^{-2} \log \delta^{-1})$.*

▶ **Theorem 2.** *[29] Given $\varepsilon, \delta \in (0, 1)$, there is an algorithm that with probability at least $1-\delta$ that $(1\pm\varepsilon)$-approximates $\|x\|_0$. It uses space $\mathcal{O}(\varepsilon^{-2} \log(\delta^{-1}) \log(n)(\log(1/\varepsilon) + \log\log(mM)))$, and its update and reporting time are both in $\mathcal{O}(\log \delta^{-1})$.*

Given $k \leq n$, x is k-sparse if it has at most k non-zero coordinates.

▶ **Theorem 3.** *[11] Given k and error probability $\delta \in (0, 1)$, there exists an algorithm that recovers a k-sparse vector exactly with probability $1 - \delta$ with space usage $\mathcal{O}(k \log n \cdot \log(k/\delta))$.*

The heavy hitter problem seeks all coordinates from a frequency vector with value greater than a certain threshold. In this work, we focus on a variant of this problem where we have to output all coordinates i such that $|x_i| \geq \varepsilon \|x\|_1$. Cormode and Muthukrishnan [13] obtained the following theorem, based on Count-Min sketch.

▶ **Theorem 4.** *[13] Given $\varepsilon, \tau \in (0, 1)$, there is an algorithm that outputs all items with frequency at least $(\varepsilon + \tau)\|x\|_1$, and with probability $1 - \delta$ outputs no items with frequency less than $\tau\|x\|_1$. It has update time $\mathcal{O}(\log n \cdot \log(2 \log n/(\delta\tau)))$ and query time $\mathcal{O}(\varepsilon^{-1})$, and uses $\mathcal{O}(\varepsilon^{-1} \log n \cdot \log(2 \log n/(\delta\tau)))$ space.*

3 Estimating Caro-Wei Bound in Average-Degree Graphs

In this section, we estimate the independence number, β, in graphs with bounded average degree. There exists a folklore offline greedy algorithm that in expectation outputs an independent set of size the Caro-Wei Bound, λ. Given a permutation of the vertices chosen uniformly at random, the greedy algorithm adds a vertex to the solution whenever it is earlier in the permutation than all its neighbors. Importantly, storing such a permutation explicitly requires $\Omega(n \log n)$ bits, which breaks our space requirement.

We overcome this by simulating a random permutation with a hash function drawn randomly from an ε-min-wise hash function family, $\mathcal{H}$. To generate a pseudo-permutation, we order vertices by their hash values. Only a vertex whose hash value is lower than that of all its neighbors is a candidate to be in the solution. By the property of the ε-min-wise hash, the probability that each vertex has smallest hash is approximately proportional to the reciprocal of its neighborhood size. In general, the neighborhood might be large, i.e., $\Theta(n)$, so we rely on standard sampling and recovery techniques in our algorithm, Caraway.

We choose the output universe of $\mathcal{H}$ in Caraway to be $[n^3]$ for two reasons. First, with high probability, there are no colliding pairs. Second, assuming $\varepsilon \in \omega(n^{-2})$ ensures that the ε-min-wise property can be applied to every subset of size at least n.

Algorithm 1 Caraway, estimating Caro-Wei bound

1: **Input**: Average degree, $\bar{d}$, error rate, ε.
2: **Initialization**: Uniformly-at-random select $h \in \mathcal{H}$, an ε-min-wise hash from $[n]$ to $[n^3]$
3: $p \leftarrow 4(\bar{d}+1)/(\varepsilon^2 n)$
4: $S \leftarrow pn$ vertices sampled uniformly at random from $[n]$
5: **for all** $e = (u, v)$ in the stream **do**
6:     **if** $(u \in S) \wedge (h(u) \geq h(v))$ **then**
7:         Remove u from S
8:     Perform the same operation on v
9: **return** $\widehat{\lambda} \leftarrow |S|/p$

▶ **Theorem 5.** *Given $\varepsilon \in (0,1)$, for every graph with average degree $\bar{d}$ and max degree $\Delta \leq \varepsilon^2\, n/(3\,(\bar{d}+1)^3)$, Algorithm 1 is an insertion-only algorithm that $(1 \pm \varepsilon)$-approximates λ. It needs space $\mathcal{O}(\bar{d}\,\varepsilon^{-2}\log n)$ and succeeds with probability at least $2/3$.*

To prove Theorem 5, we start with a result on the expectation of $\widehat{\lambda}$.

▶ **Lemma 6.** [$\star$][6] *$E[\widehat{\lambda}] = (1 \pm \varepsilon)\,\lambda$.*

We next show that, with constant probability, $\widehat{\lambda}$ is no more than a $1 \pm \varepsilon$ factor away from $E[\widehat{\lambda}]$. Let X_v be a indicator random variable for the event that v is both sampled and not removed from S. Let $X = \sum_{v \in V} X_v$. Since some pairs of X_v and X_u are positively correlated, we cannot directly apply a Chernoff bound. There are three cases to consider.

- Vertices u and v are adjacent: $u \in N[v]$ and $v \in N[u]$. If X_u is 1, wlog, we know that X_v is 0. Hence X_v and X_u are negatively correlated.
- Vertices u and v are not adjacent, but share at least one neighbor. Consider one such common neighbor, w, i.e., $w \in N[u] \cap N[v]$. Knowing u has the smallest hash value in $N[u]$, i.e., $h(w) > h(u)$, implies $h(v) < h(w)$ is likely, and hence v is more likely to have the smallest hash value in $N[v]$. Random variables X_v and X_u are positively correlated. Lemma 7 consolidates this positive correlation.
- Vertices u and v do not fall into the previous two cases: X_v and X_u are independent (as confirmed by Lemma 7).

▶ **Lemma 7.** [$\star$] *Let x and y be non-adjacent vertices with $|N(x) \cap N(y)| = k$, $|N(x) \setminus N(y)| = l$, and $|N(y) \setminus N(x)| = r$. If the vertices are mapped to a random permutation, we have* $\Pr[y < N(y) \mid x < N(x)] = \frac{(l+r+2k+2)}{(r+k+1)(l+k+r+2)}$.

[6] Due to space constraints, proofs of all results labeled [$\star$] are deferred to the Appendix.

Within a factor $1 \pm \varepsilon$, as shown in Appendix Section B.2, in a permutation generated from a hash function from a family with the ε-min-wise property, Lemma 7 also holds: it relies only the probability of an element being the smallest.

By Lemma 7, $\Pr[y < N(y) \mid x < N(x)] = 1/(r+1) = \Pr[y < N(y)]$ when $k = 0$: x and y have no common neighbor. However, if $k > 0$, then $\frac{(l+r+2k+2)}{(r+k+1)(l+k+r+2)} > \frac{1}{r+k+1}$. So $x < N(x)$ and $y < N(y)$ are positively correlated if and only if x and y share at least one neighbor.

With a Chernoff-type bound unavailable, we bound the variance of $\widehat{\lambda}$, and apply Chebyshev's inequality. It suffices to bound the variance of X as $\mathrm{Var}(\widehat{\lambda}) = \mathrm{Var}(|S|/p) = \mathrm{Var}(X)$.

▶ **Lemma 8.** [⋆] *If $\Delta \leq \varepsilon^2 n/(3(\bar{d}+1)^3)$ and $p \geq 4(\bar{d}+1)/(\varepsilon^2 n)$, then $Var(X) \leq \varepsilon^2 E^2[X]/3$.*

Now we have all the tools to prove Theorem 5, see Appendix Section B.4 for the proof. By running several instances and returning the median, the success probability of Algorithm 1 is boosted by to $1 - \delta$ in $\mathcal{O}(\bar{d}\varepsilon^{-2} \log \delta^{-1} \cdot \log n)$ space.

3.1 Extension 1: An Independent Set in the Online Streaming model

Algorithm 1 only estimates the Caro-Wei bound, λ. It can be easily modified to output an actual independent set in the *online streaming* model. Starting with an n-bit array set to true for the solution, the working space of our algorithm is merely the space to store the hash function, i.e., $\mathcal{O}(\log \varepsilon^{-1} \log n)$. Moreover, the update time of our algorithm is the time to compute a hash value, $\mathcal{O}(\log \varepsilon^{-1})$. This result is summarised in the following theorem.

▶ **Theorem 9.** [⋆] *For graphs with average degree $\bar{d}$ and max degree $\Delta \leq \varepsilon^2 n/(3(\bar{d}+1)^3)$, there is an* online streaming *algorithm that, with probability at least $2/3$, reports an independent set of size $1 \pm \varepsilon$ times the Caro-Wei bound. Working space is $\mathcal{O}(\log \varepsilon^{-1} \cdot \log n)$ and update time is $\mathcal{O}(\log \varepsilon^{-1})$.*

3.2 Extension 2: Removing the Constraint on Max Degree

The bound $\frac{\varepsilon^2\, n}{3\,(\bar{d}+1)^3}$ on the maximum degree, Δ, is a requirement for Theorem 5. We can avoid this constraint by ignoring the heavy hitters (i.e., high-degree vertices) in the graph, without significantly changing λ, as formalized in Lemma 10. Recall that H_μ is the set of vertices with degree at least μ and that G_{-H_μ} is the induced subgraph on $V \setminus H_\mu$.

▶ **Lemma 10.** [⋆] *For every $\mu \geq (\bar{d}+1)/\sqrt{\varepsilon}$, we have $\lambda(G) \geq \lambda(G_{-H_\mu}) \geq (1-\varepsilon)\lambda(G)$.*

Let R be the vertices to be ignored, which should [7] be those with degree at least $(\bar{d}+1)/\sqrt{\varepsilon}$. Ignoring R, similar to Algorithm 1, Subroutine 2 approximates λ in the remaining graph by sampling vertices and removing some based on hash values. By Lemma 10, this estimate is at most factor $1 - \varepsilon$ away from the Caro-Wei bound of the original graph, G.

We identify the high-degree vertices, R, via the heavy hitter sketch with careful parameter settings (see Section 3.2.1). Since R is only reported at the end of stream, Subroutine 2 retains all neighbors of vertices in S, deferring the heavy-hitter ignoring and hash-value removal to post-processing. If two passes are allowed, or if R is provided in advance, then these two steps can be done on-the-fly, fitting into the online streaming model. In particular, the condition in line 6 of Algorithm 1 is changed to $(u \in S) \wedge (v \notin R) \wedge (h(u) \geq h(v))$.

[7] Lemma 11 clarifies exactly which vertices are in R.

Subroutine 2 Caro-Wei bound without Degree Constraint

1: **Input 1**: the average degree, $\bar{d}$, the error rate, ε
2: **Initialization**: Uniformly-at-random select $h \in \mathcal{H}$, an ε-min-wise hash from $[n]$ to $[n^3]$
3: $p \leftarrow 4(\bar{d}+1)/(\varepsilon^2 n)$
4: $S \leftarrow pn$ vertices sampled uniformly at random from $[n]$
5: **for** $u \in S$ **do**
6: $\quad L_u \leftarrow \{\}$
7: **for all** $e = (u, v)$ in the stream **do**
8: $\quad$ **if** $u \in S$ **then**
9: $\quad\quad$ Add v to L_u
10: $\quad$ Perform the same operation on v
11: $\quad$ **if** $\sum_{u \in S}(1 + |L_u|) > 10\bar{d}pn$ **then**
12: $\quad\quad$ **Abort**

1: **Post-processing**:
2: **Input 2**: a set of ignored vertices, R
3: **for all** $v \in R$ **do**
4: $\quad$ **for all** $u \in S$ **do**
5: $\quad\quad$ $L_u \leftarrow L_u \setminus \{v\}$
6: **for all** $v \in S$ **do**
7: $\quad$ **for all** $u \in L_v$ **do**
8: $\quad\quad$ **if** $h(u) < h(v)$ **then**
9: $\quad\quad\quad$ $S \leftarrow S \setminus \{v\}$
10: **return** $\widehat{\lambda} = |S|/p$

▶ **Lemma 11.** [$\star$] *Suppose R contains all vertices with degree at least $\frac{\varepsilon^2}{3} \frac{n}{(\bar{d}+1)^3}$, but no vertices with degree less than $(\bar{d}+1)/\sqrt{\varepsilon}$. Given $\varepsilon \in (0, 1)$, for every graph with average degree $\bar{d}$, Subroutine 2 is an insertion-only algorithm that $(1-\varepsilon)(1 \pm \varepsilon)$-approximates the Caro-Wei bound. It uses space $\mathcal{O}(\bar{d}^2\varepsilon^{-2} \log n)$ and succeeds with probability at least* 19/30.

3.2.1 Identifying heavy hitters

≪Tony: I think we could make this subsubsection much shorter!≫ The success of Subroutine 2 depends on identifying the set of high-degree vertices, R, well. By Theorem 4, running the algorithm of Cormode and Muthukrishnan [13] with $\varepsilon = \tau = \frac{\varepsilon^2}{6(\bar{d}+1)^4}$ and $\delta = n^{-c}$, for some constant c, we obtain the desired removal set with high probability. This heavy-hitter technique requires $\mathcal{O}(\varepsilon^{-1} \log^2 n)$ space. By running $c' \log n$ (for some constant c') instances of Subroutine 2 concurrently, and returning the maximum result, the failure probability drops to $n^{-c'}$. Importantly, there is only one instance of the heavy-hitter algorithm: it feeds its output R to every instance of Subroutine 2. Via a union bound, the whole process's failure probability is at most n^{-c}, hence:

▶ **Theorem 12.** *Given $\varepsilon \in (0, 1)$, for every graph with average degree $\bar{d}$, there is an algorithm that whp $(1-\varepsilon)(1 \pm \varepsilon)$-approximates the Caro-Wei bound in $\mathcal{O}(\bar{d}^2\varepsilon^{-2} \log^2 n)$ space.*

3.3 Vertex-Arrival Random-Order Algorithm

≪Tony: And this one!≫ So far, we have relied on a hash function drawn from an ε-min-wise hash family to approximate a random permutation. However, if the stream is vertex-arrival and random-order, the random permutation comes for free. Moreover, we can immediately tell whether the vertex is earliest in its neighborhood because the convention is only to list prior-occurring neighbors. Hence, it suffices to count the number of such vertices, with space further reduced to $\mathcal{O}(\log(\bar{d}\varepsilon^{-2}))$ by incrementing the counter with probability $p = \frac{4(\bar{d}+1)}{\varepsilon^2 n}$. We abort the algorithm if $c > 10pn$, which happens with probability less than 1/10. We conclude the following theorem:

▶ **Theorem 13.** [⋆] *Given $\varepsilon \in (0,1)$, for every graph with average degree $\bar{d}$ and max degree $\Delta \leq \frac{\varepsilon^2 n}{3(\bar{d}+1)^3}$, there is an insertion-only random-order vertex-arrival algorithm that $(1 \pm \varepsilon)$-approximates λ. It succeeds with probability at least $19/30$ in space $\mathcal{O}(\log(\bar{d}\varepsilon^{-2}))$.*

4 Estimating Graph Parameters in Forests

In this section, we describe our streaming algorithms when the input graph is a forest. Unless otherwise stated, we assume that the input forest has no isolated vertices. Our algorithms involve a linear combination of several combinatorial graph parameters, in particular support vertices. Estimating these in small space leads to effective streaming graph algorithms.

4.1 Combinatorial Graph Parameters

Leaves and Non-leaf Vertices: The number of leaves and non-leaf vertices are helpful parameters for approximating the matching number in forests [6, 18] and domination number in trees [17, 35, 32]. For a forest, F, or tree, T, let Deg_1 be the set of leaves, c be the number of tree components of F, and $Deg_{\geq 2}$ be the set of non-leaf vertices. These two known results bound the matching number and domination number of forests in terms of the aforementioned graph parameters. ≪Tony: Perhaps a sentence or two about why they are or are not useful.≫

▶ **Theorem 14.** *[6, 18] For every forest F,* $\max\left\{ c\,,\, \frac{1}{2}(|Deg_{\geq 2}| + c) \right\} \leq \phi \leq |Deg_{\geq 2}| + c\,.$

▶ **Theorem 15.** *[17, 35, 32] For every tree T with $n \geq 3$,* $\frac{1}{3}\,|Deg_{\geq 2}| \leq \gamma \leq |Deg_{\geq 2}|\,.$

In Theorem 15, constraint $n \geq 3$ accounts for the case of a tree with $n = 2$ vertices, a single edge: each vertex is a leaf, but $\gamma = 1$. If the upper bound is relaxed by 1, all trees are included, leading to the forest generalization of Theorem 15:

▶ **Theorem 16.** *For every forest F,* $\frac{1}{3}\,|Deg_{\geq 2}| \leq \gamma \leq |Deg_{\geq 2}| + c\,.$

≪Tony: Now we turn to our results. We first start with something about β.≫
With no known bound for independence number, β, in a forest F via c and $|Deg_1|$, we show:

▶ **Theorem 17.** [⋆] *For every forest F,* $\max\left\{ \frac{1}{2}\,n\,,\, |Deg_1| - c\right\} \leq \beta \leq \frac{1}{2}\big(n + |Deg_1|\big)\,.$

Support Vertices: We here demonstrate the power of support vertices, i.e., vertices adjacent to leaves. Let *Supp* be the set of support vertices in forest F.

≪Tony: The purpose of these is to introduce support vertices: but perhaps the corollaries are introduced, with the constituent theorems only appearing in the appendix≫

▶ **Theorem 18.** [⋆] *For every forest F,* $\frac{1}{2}\big(n+|Deg_1|-|Supp|\big) \leq \beta \leq \frac{2}{3}\big(n+|Deg_1|-|Supp|\big)\,.$

▶ **Theorem 19.** [⋆] *For every forest F,* $\frac{1}{4}\big(|Deg_{\geq 2}| + |Supp|\big) \leq \gamma \leq \frac{1}{2}\big(|Deg_{\geq 2}| + |Supp|\big)\,.$

▶ **Theorem 20.** [⋆] *For every forest F,* $\frac{1}{3}\big(|Deg_{\geq 2}| + |Supp|\big) \leq \phi \leq \frac{1}{2}\big(|Deg_{\geq 2}| + |Supp|\big)\,.$

Combining these three theorems with the theorems just above, we achieve these corollaries.

▶ **Corollary 21.** [⋆] *If $|Supp| \leq \frac{1}{2}\,|Deg_{\geq 2}|$, then* $\frac{3}{8}\big(n + |Deg_1|\big) \leq \beta \leq \frac{1}{2}\big(n + |Deg_1|\big)\,.$

▶ **Corollary 22.** [⋆] *If $|Supp| \leq \frac{1}{3}\,|Deg_{\geq 2}|$, then* $\frac{1}{3}\,|Deg_{\geq 2}| \leq \gamma \leq \frac{2}{3}\,|Deg_{\geq 2}|\,.$

▶ **Corollary 23.** [⋆] *If $|Supp| \leq \frac{1}{2}\,|Deg_{\geq 2}|$, then* $\max\left\{c, \frac{|Deg_{\geq 2}|+c}{2}\right\} \leq \phi \leq \frac{3}{4}\cdot\big(|Deg_{\geq 2}| + c\big)\,.$

4.2 Streaming Algorithms

The number of trees, $c(F)$, in a forest F is $n-m$; storing this requires only $\mathcal{O}(\log n)$ bits.

Non-leaf Vertices: By updating the entries relevant to the newly arrived elements in the stream, we easily construct the degree vector, $\mathcal{D}(G)$. The quantity $|Deg_{\geq 2}(G)|$ is exactly the number of coordinates in $\mathcal{D}(G)$ with value at least 2. Since we assume no isolated vertices, by subtracting the all-ones vector $\{1\}^n$ from $\mathcal{D}(G)$, i.e., initializing $\mathcal{D}'(G) \leftarrow \mathcal{D}(G) + \{-1\}^n$, the number of non-zero entries, $L_0(\mathcal{D}'(G))$, is exactly $|Deg_{\geq 2}(G)|$. Hence, an algorithm that $(1 \pm \varepsilon)$-approximates $\|\mathcal{D}'(G)\|_0$ gives an $(1 \pm \varepsilon)$ approximation to $|Deg_{\geq 2}(G)|$. Since no existing L_0 sketch supports initialization with $\{-1\}^n$, degree decrements are post-processed.

By Theorem 2, with constant success probability, we $(1 \pm \varepsilon)$-approximate $|Deg_{\geq 2}(G)|$ in $\log^{\mathcal{O}(1)} n$ space. We boost the success probability to $(1-\delta)$ by running independent instances and returning the median, hence:

▶ **Lemma 24.** *For every $\varepsilon, \delta \in (0,1)$ and graph G, $|Deg_{\geq 2}|$ can be $(1 \pm \varepsilon)$-approximated in a turnstile stream with probability $(1-\delta)$ and in $\mathcal{O}(\log^{\mathcal{O}(1)} n \cdot \log \delta^{-1})$ space.*

Combining Lemma 24 with Theorem 14 and Theorem 16, respectively, we have:

≪Tony: How could we make this easier to read? With a table?≫

▶ **Theorem 25.** *For every $\varepsilon, \delta \in (0,1)$ and forest F, the matching number ϕ can be $(2 \pm \varepsilon)$-approximated in turnstile streams with probability $(1-\delta)$ and in $\mathcal{O}(\log^{\mathcal{O}(1)} n \cdot \log \delta^{-1})$ space.*

▶ **Theorem 26.** *For every $\varepsilon, \delta \in (0,1)$ and forest F, the domination number γ can be $(3 \pm \varepsilon)$-approximated in turnstile streams with probability $(1-\delta)$ and in $\mathcal{O}(\log^{\mathcal{O}(1)} n \cdot \log \delta^{-1})$ space.*

Leaves: We could estimate $|Deg_1(F)|$ in a forest F with no isolated vertices by the expression $n - |Deg_{\geq 2}(F)|$. But if $|Deg_1(F)|$ is $o(|Deg_{\geq 2}(F)|)$ it would be inaccurate. We turn elsewhere:

▶ **Lemma 27.** [⋆] *For every forest F, we have $|Deg_1(F)| = 2c(F) + \sum_{i=3}^{\Delta}(i-2) \cdot |Deg_i(T)|$.*

Now, subtracting 2 from each coordinate of degree vector $\mathcal{D}(F)$ (i.e., initializing $\mathcal{D}''$ with $\{-2\}^n$ rather than $\{0\}^n$), we have $\|\mathcal{D}''(F)\|_1 = |Deg_1(F)| + \sum_{i=3}^{\Delta}(i-2) \cdot |Deg_i(F)|$. Folding in Lemma 27, we have $|Deg_1(F)| = L_1(\mathcal{D}''(F))/2 + c(F)$. As before, we postpone degree double decrements to a post processing. By Theorem 1, $|Deg_1(F)|$ is $(1 \pm \varepsilon)$-approximated with constant success probability in $\log^{\mathcal{O}(1)} n$ space; with the usual trick, we have:

▶ **Lemma 28.** *For every $\varepsilon, \delta \in (0,1)$ and forest F with n vertices, $|Deg_1(F)|$ can be $(1 \pm \varepsilon)$-approximated in a turnstile stream with probability $(1-\delta)$ in $\mathcal{O}(\log^{\mathcal{O}(1)} n \cdot \log \delta^{-1})$ space.*

Combining Lemma 28 with Theorem 17, we conclude:

▶ **Theorem 29.** *The independence number of forests on n vertices can be $3/2 \cdot (1 \pm \varepsilon)$-approximated with probability $(1-\delta)$ on turnstile streams using $\mathcal{O}(\log^{\mathcal{O}(1)} n \cdot \log \delta^{-1})$ space.*

Support Vertices: ≪Tony: Slightly odd sentence≫ Support vertices are powerful, so we show, given a forest F in a turnstile stream, how to output an $(1 \pm \varepsilon)$-estimate of $|Supp(F)|$ when $|Supp(F)|$ is at least threshold K_1. Similar to Cormode et al. [12] and to Jayaram and Woodruff [26], we hence assume the number of deletions $\mathcal{O}(n)$. If an arbitrary number of deletions were allowed, the number of insertions might be arbitrarily large, breaking our analyses of Subroutine 3 and Subroutine 4.

≪Tony: Can we wrap text around this algorithm?≫

Subroutine 3 Estimating $|Supp(F)|$ for a forest F

Input: A size threshold K_1, a constant c_1, and error rate $\varepsilon_1 \in (0,1)$
Initialization: $I \leftarrow \frac{c_1 n}{\varepsilon_1^2 K_1}$ vertices sampled uniformly at random from $[n]$
for $v \in I$ **do**
    $l(v) \leftarrow \{\}$
$m, t \leftarrow 0$
First Pass:
for all $(e = (u, v), i)$ in the stream, with $i \in \pm 1$ **do**
    $m \leftarrow m + i$
    if $u \in I$ **then**
        Toggle v's presence in $l(u)$
        $t \leftarrow t + i$
    Perform the same operation on v
    Abort if $t \geq \frac{2m}{n} \frac{c_1 n}{\varepsilon_1^2 K_1} e^{\frac{c_1}{3}}$
Second Pass:
Count the degree of every vertex in $I \cup \left(\bigcup_{w \in I} l(w)\right)$
$C \leftarrow \{u \mid u \in I$ and there is some leaf $v \in l(u)\}$
return $|\widehat{Supp(F)}| \leftarrow |C| \times \frac{\varepsilon_1^2 K_1}{c_1}$

▶ **Lemma 30.** [⋆] *Given size threshold K_1, constant c_1, and error rate $\varepsilon_1 \in (0,1)$, Subroutine 3 is a turnstile stream algorithm that uses $\widetilde{\mathcal{O}}(\frac{n}{\varepsilon_1^2 K_1})$ space and it*

- *For $|Supp(F)| \geq K_1$, $(1 \pm \varepsilon_1)$-approximates $|Supp(F)|$ with probability at least $1 - 3e^{-\frac{c_1}{3}}$.*
- *For $|Supp(F)| < K_1$, returns an estimate at most $2K_1$ with probability at least $1 - 2e^{-\frac{c_1}{3}}$.*

Again letting $\mathcal{D}'(F) = \mathcal{D}(F) + \{-1\}^n$, when $|Deg_{\geq 2}(F)|$ is small, sparse recovery on $\mathcal{D}'(F)$ recovers all vertices in $Deg_{\geq 2}(F)$, with probability $1 - \delta$, but no leaves. In a second pass, we verify whether a recovered vertex is a support vertex: except for a length-two path (a P_2), the condition is whenever some neighbor is not recovered. An edge is a P_2 if neither endpoint is recovered. To verify whether we have successfully recovered all vertices in $Deg_{\geq 2}(F)$, we sum degrees, knowing all leaves are unrecovered. The full algorithm is described in Subroutine 4.

▶ **Lemma 31.** [⋆] *Given size threshold K_2, constant c_2, and a forest F as a turnstile stream, Subroutine 4 in two passes outputs exact estimates for both $|Supp(F)|$ and $|Deg_{\geq 2}(F)|$ in $\widetilde{\mathcal{O}}(K_2)$ space. When $|Deg_{\geq 2}(F)| \leq K_2$, the failure probability is at most $1/c_2$.*

Finalizing the Algorithms: In our two-pass Algorithm 5 for estimating the independence number in a forest, we run Subroutine 3 and Subroutine 4 concurrently, returning the minimum of $(3(n + |Deg_1(F)|))/8$ and $(n + |Deg_1(F)| - |Supp(F)|)/2$.

▶ **Theorem 32.** [⋆] *For every $\varepsilon, \delta \in (0,1)$ and forest F (with no isolated vertices), Algorithm 5 estimates the independence number $\beta(F)$ in a turnstile stream within factor $4/3 \cdot (1 \pm \varepsilon)$ with probability $(1 - \delta)$ in $\widetilde{\mathcal{O}}(\sqrt{n})$ space and two passes.*

Two-pass algorithms for domination number and matching are similar.

▶ **Theorem 33.** *For every $\varepsilon, \delta \in (0,1)$ and forest F (with no isolated vertices), Algorithm 5 estimates the domination number $\gamma(F)$ in a turnstile stream within factor $2 \cdot (1 \pm \varepsilon)$ with probability $(1 - \delta)$ in $\widetilde{\mathcal{O}}(\sqrt{n})$ space and two passes.*

▶ **Theorem 34.** *For every* $\varepsilon, \delta \in (0,1)$ *and forest* F *(with no isolated vertices), Algorithm 5 estimates the matching number* $\phi(F)$ *in a turnstile stream within factor* $3/2 \cdot (1 \pm \varepsilon)$ *with probability* $(1-\delta)$ *in* $\widetilde{\mathcal{O}}(\sqrt{n})$ *space and two passes.*

Subroutine 4 Estimating $|Supp(F)|$ when $|Deg_{\geq 2}(F)| \leq K_2$

Input: A size threshold K_2, a large constant c_2
Initialization: Sparse recovery sketch $\mathcal{L}$ in [11] with $k = K_2$ and $\delta = 1/c_2$
First Pass:
for all $(e = (u,v), i)$ in the stream, $i \in \pm 1$ **do**
    Update $\mathcal{L}$ with (e, i)
Decrement all coordinates in $\mathcal{L}$ by 1
Denote the result of sparse recovery from $\mathcal{L}$ by $\mathcal{R}$
$p, m \leftarrow 0$
for $v \in \mathcal{R}$ **do**
    $d_v, l_v \leftarrow 0$ ▷ Degree and leaf counters
Second Pass:
for all $(e = (u,v), i)$ in the stream, $i \in \pm 1$ **do**
    $m \leftarrow m + i$; $d_u \leftarrow d_u + i$; $d_v \leftarrow d_v + i$
    if $u \notin \mathcal{R}$ and $v \notin \mathcal{R}$ **then**
        $p \leftarrow p + i$
    if $u \in \mathcal{R}$ and $v \notin \mathcal{R}$ **then**
        $l_u \leftarrow l_u + i$
    if $v \in \mathcal{R}$ and $u \notin \mathcal{R}$ **then**
        $l_v \leftarrow l_v + i$
if $2m \neq n - |\mathcal{R}| + \sum_{v \in \mathcal{R}} d_v$ **then**
    Return *FAIL*
else
    $|\widehat{Supp(F)}| \leftarrow |\{u \mid u \in \mathcal{R} \text{ and } l(u) = 1\}| + 2p$
    $|\widehat{Deg_{\geq 2}(F)}| \leftarrow |\mathcal{R}|$
    Return $|\widehat{Supp(F)}|$ and $|\widehat{Deg_{\geq 2}}|$

Algorithm 5 Estimating $\beta(F)$ in forest F

1: **Input**: error rate ε, fail rate δ
2: **Initialization**: $K_1 \leftarrow \sqrt{n}$, $K_2 \leftarrow 8\sqrt{n}$, $c_1 \leftarrow 3\ln(6\delta^{-1})$, $c_2 \leftarrow \delta^{-1}$, $\varepsilon_1 \leftarrow \varepsilon/2$
3: Run the following three subroutines concurrently:
4: $|\widehat{Deg_1}| \leftarrow$ Lemma 28 algorithm with $\varepsilon/2$ and $\delta/2$
5: $|\widehat{Supp(F)}| \leftarrow$ Subroutine 3 with K_1, c_1, and ε_1
6: $(|\widehat{Supp(F)}|', |\widehat{Deg_{\geq 2}}|') \leftarrow$ Subroutine 4 with K_2 and c_2
7: **if** Subroutine 4 returns $FAIL$ **then**
8:     **Return** $\widehat{\beta} \leftarrow \min\{\frac{3(n+|\widehat{Deg_1}|)}{8}, \frac{n+|\widehat{Deg_1}|-|\widehat{Supp(F)}|}{2}\}$
9: **else**
10:     $|\widehat{Deg_1}|' \leftarrow n - |\widehat{Deg_{\geq 2}}|'$
11:     **Return** $\widehat{\beta} \leftarrow \min\{\frac{3(n+|\widehat{Deg_1}|')}{8}, \frac{n+|\widehat{Deg_1}|'-|\widehat{Supp(F)}|'}{2}\}$

References

1 Filipe Araujo, Jorge Farinha, Patricio Domingues, Gheorghe Cosmin Silaghi, and Derrick Kondo. A maximum independent set approach for collusion detection in voting pools. *Journal of Parallel and Distributed Computing*, 71(10):1356–1366, 2011.

2 Sepehr Assadi, Sanjeev Khanna, and Yang Li. Tight bounds for single-pass streaming complexity of the set cover problem. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC 2016)*, pages 698–711. ACM, 2016.

3 Jana Bauckmann, Ziawasch Abedjan, Ulf Leser, Heiko Müller, and Felix Naumann. Discovering conditional inclusion dependencies. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012)*, pages 2094–2098. ACM, 2012.

4 Ravi B Boppana, Magnús M Halldórsson, and Dror Rawitz. Simple and local independent set approximation. In *Proceedings of the 25th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2018)*, pages 88–101. Springer, 2018.

5 Marc Bury, Elena Grigorescu, Andrew McGregor, Morteza Monemizadeh, Chris Schwiegelshohn, Sofya Vorotnikova, and Samson Zhou. Structural results on matching estimation with applications to streaming. *Algorithmica*, 81(1):367–392, 2019.

6 Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA 2015)*, pages 263–274. Springer, 2015.

7 Yair Caro. New results on the independence number. Technical report, Technical Report, Tel-Aviv University, 1979.

8 Rajesh Chitnis and Graham Cormode. Towards a theory of parameterized streaming algorithms. In *Proceedings of the 14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

9 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 1326–1344. SIAM, 2016.

10 Graham Cormode, Jacques Dark, and Christian Konrad. Approximating the Caro-Wei bound for independent sets in graph streams. In *Proceedings of the 5th International Symposium on Combinatorial Optimization (ISCO 2018)*, pages 101–114. Springer, 2018.

11 Graham Cormode and Donatella Firmani. A unifying framework for ℓ_0-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335, 2014.

12 Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In

Proceedings of the 25th Annual European Symposium on Algorithms (ESA 2017), page 29. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

13 Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

14 Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. Finding related tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2012)*, pages 817–828. ACM, 2012.

15 Ermelinda DeLaVina, Craig E Larson, Ryan Pepper, Bill Waller, and Odile Favaron. On total domination and support vertices of a tree. *AKCE International Journal of Graphs and Combinatorics*, 7(1):85–95, 2010.

16 Dong Deng, Raul Castro Fernandez, Ziawasch Abedjan, Sibo Wang, Michael Stonebraker, Ahmed Elmagarmid, Ihab F Ilyas, Samuel Madden, Mourad Ouzzani, and Nan Tang. The data civilizer system. In *Proceedings of the 8th Biennial Conference on Innovative Data Systems Research (CIDR 2017)*, 2017.

17 Stephan J Eidenbenz. Online dominating set and variations on restricted graph classes. *Technical Report/ETH Zurich, Department of Computer Science*, 380, 2002.

18 Hossein Esfandiari, Mohammad T Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 1217–1233. SIAM, 2015.

19 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.

20 Venkatesh Ganti and Anish Das Sarma. Data cleaning: A practical perspective. *Synthesis Lectures on Data Management*, 5(3):1–85, 2013.

21 Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. Evaluation of labeling strategies for rotating maps. *Journal of Experimental Algorithmics (JEA)*, 21:1–21, 2016.

22 Bjarni V Halldórsson, Magnús M Halldórsson, Elena Losievskaja, and Mario Szegedy. Streaming algorithms for independent sets in sparse hypergraphs. *Algorithmica*, 76(2):490–501, 2016.

23 Magnús M Halldórsson and Jaikumar Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18(1):145–163, 1997.

24 Ayaan Hossain, Eriberto Lopez, Sean M Halper, Daniel P Cetnar, Alexander C Reis, Devin Strickland, Eric Klavins, and Howard M Salis. Automated design of thousands of nonrepetitive parts for engineering stable genetic systems. *Nature Biotechnology*, 38(12):1466–1475, 2020.

25 Piotr Indyk. A small approximately min-wise independent family of hash functions. *Journal of Algorithms*, 38(1):84–90, 2001.

26 Rajesh Jayaram and David P Woodruff. Data streams with bounded deletions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2018)*, pages 341–354. ACM, 2018.

27 David S Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

28 Daniel M Kane, Jelani Nelson, and David P Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 1161–1178. SIAM, 2010.

29 Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2010)*, pages 41–52. ACM, 2010.

30 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

31 Tim Kieritz, Dennis Luxen, Peter Sanders, and Christian Vetter. Distributed time-dependent contraction hierarchies. In *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA 2010)*, pages 83–93. Springer, 2010.

32 Magdalena Lemańska. Lower bound on the domination number of a tree. *Discussiones Mathematicae Graph Theory*, 24(2):165–169, 2004.

33 László Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, 1975.

34 Andrew McGregor and Sofya Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In *Proceedings of the 1st Symposium on Simplicity in Algorithms (SOSA 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

35 Dirk Meierling and Lutz Volkmann. A lower bound for the distance k-domination number of trees. *Results in Mathematics*, 47(3-4):335–339, 2005.

36 Tijana Milenković, Vesna Memišević, Anthony Bonato, and Nataša Pržulj. Dominating biological networks. *PLOS One*, 6(8):0023016, 2011.

37 Morteza Monemizadeh, S. Muthukrishnan, Pan Peng, and Christian Sohler. Testable bounded degree graph properties are random order streamable. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, pages 131:1–131:14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

38 Jose C Nacher and Tatsuya Akutsu. Dominating scale-free networks with variable scaling exponent: Heterogeneous networks are not difficult to control. *New Journal of Physics*, 14(7):073005, 2012.

39 O. Ore. *Theory of Graphs*. American Mathematical Society Colloquium Publications. American Mathematical Society, 1962.

40 Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

41 Pan Peng and Christian Sohler. Estimating graph parameters from random order streams. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 2449–2466. SIAM, 2018.

42 Tayler Pino, Salimur Choudhury, and Fadi Al-Turjman. Dominating set algorithms for wireless sensor networks survivability. *IEEE Access*, 6:17527–17532, 2018.

43 Chao Shen and Tao Li. Multi-document summarization via the minimum dominating set. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 984–992. ACM, 2010.

44 Paul Turán. On an extremal problem in graph theory. *Mat. Fiz. Lapok*, 48(436-452):137, 1941.

45 Jiannan Wang, Guoliang Li, and Jianhua Fe. Fast-join: An efficient method for fuzzy token matching based string similarity join. In *Proceedings of the 27th International Conference on Data Engineering (ICDE 2011)*, pages 458–469. IEEE, 2011.

46 VK Wei. A lower bound on the stability number of a simple graph. Technical report, Bell Laboratories Technical Memorandum, 1981.

47 Jiguo Yu, Nannan Wang, Guanghui Wang, and Dongxiao Yu. Connected dominating sets in wireless ad hoc and sensor networks–a comprehensive survey. *Computer Communications*, 36(2):121–134, 2013.

A Supplementary Preliminaries

A.1 The Data Stream Model

A graph stream is *edge-arrival* if edges arrive one by one, sequentially. If vertices arrive sequentially together with their neighbors that have arrived earlier, it is *vertex-arrival*. Also, the stream is *insertion-only* if there are no deletions. While in *turnstile* streams, an element can be deleted if it has been inserted and not been deleted. Moreover, the order of arrival can be either *arbitrary* or *random*.

In *online streaming* model, elements arrive in a streaming fashion, and at any given time, the algorithm must be able to report a valid solution. Usually, the algorithm will start with a initial solution, and modify it based on the newly arrived element. Similar to the *online* model, each decision on the solution is irrevocable.

A.2 ε-min-wise Hash Function

A family of hash functions, $\mathcal{H} = \{h : [n] \to [m]\}$, is ε-min-wise if for every $A \subset [n]$ and $x \in [n] \setminus A$,

$$\Pr_{h \in \mathcal{H}}[\forall a \in A, h(x) < h(a)] = \frac{1 \pm \varepsilon}{|A| + 1}.$$

Indyk [25] showed that the ε-min-wise property can be achieved via a $\mathcal{O}(\log 1/\varepsilon)$-wise hash family, under the constraints that $|A| \in \mathcal{O}(\varepsilon m)$. Its space usage is $\mathcal{O}(\log 1/\varepsilon \cdot \log m)$ and its computation time is $\mathcal{O}(\log 1/\varepsilon)$.

A.3 Extension of Chernoff-Hoeffding inequality

Panconesi and Srinivasan [40] extended the Chernoff-Hoeffding inequality to negatively correlated Boolean random variables as follows:

▶ **Theorem 35.** *[40] For r negatively correlated Boolean random variables $X_1, \ldots, X_r$, let $X = \sum_{i=1}^{r} X_i$, $\mu = E[X]$ and $0 < \delta < 1$, then*

$$\Pr\left[|X - \mu| \geq \delta\mu\right] \leq 2e^{-\frac{\mu\delta^2}{2}}.$$

B Omitted Proofs from Section 3

B.1 Proof of Lemma 6

Proof of Lemma 6. Let binary random variables X_v denote whether vertex v is sampled in S and not removed during the stream. That is, $X_v = 1$ if $v \in S$ at the end of the stream. Let $X = \sum_{v \in V(G)} X_v$, clearly, $X = |S|$ at the end. It is not hard to see that $X_v = 1$ if and only if v is sampled and v has the smallest hash value in $N[v]$. According to the property of ε-min-wise hash families, v has the smallest hash value in $N[v]$ with probability $\frac{1 \pm \varepsilon}{|N[v]|}$. By the construction of algorithm, v is sampled with probability p. Since these two events are independent, the probability that $X_v = 1$ is $(1 \pm \varepsilon)\,\frac{p}{d(v)+1}$. Hence, by the linearity of expectation, we have

$$E[|S|] = \sum_{v \in V(G)} E[X_v] = p \sum_{v \in V(G)} \frac{1 \pm \varepsilon}{d(v)+1} = (1 \pm \varepsilon)\, p\, \lambda$$

Hence, $E[\widehat{\lambda}] = E[\frac{|S|}{p}] = (1 \pm \varepsilon)\, \lambda$.

◀

B.2 Proof of Lemma 7

Proof of Lemma 7. Consider $n = l+k+r+2$ vertices $x, y, w_1, \ldots, w_l, u_1, \ldots, u_k$, and $v_1, \ldots, v_r$. For simplicity, we use W, U, and V to denote $\{w_1, \ldots, w_l\}$, $\{u_1, \ldots, u_k\}$, and $\{v_1, \ldots, v_r\}$. Note that x has degree $l+k$, y has degree $k+r$, x and y are not adjacent and they share k neighbors, i.e., $N(x) = \{W \cup U\}$ and $N(y) = \{U \cup V\}$.

We use $x < N(x)$ if x has the smallest order among $N[x]$ in the permutation. Note that since we assume uniform random permutation, $\Pr[x < N(x)] = 1/(|N(x)|+1) = 1/(l+k+1)$. Consider the event $X < N(X)$ and $Y < N(Y)$, we have two cases:

- x, followed by zero or more W, y, followed by a mixture of the rest of W, V, and U.
- y, followed by zero or more V, x, followed by a mixture of W, the rest of V, and U.

By the uniform random permutation, the *first* event occurs with probability

$$\frac{1}{l+k+r+2} \cdot \frac{1}{k+r+1},$$

where the first factor is the probability of $x < \{y\} \cup W \cup V \cup U$, and the second factor is the probability of $y < V \cup U$, regardless of how it relates with vertices in W. Note these two events are independent of each other, since knowing x being the smallest across y, W, U, V does not reveal anything about the ordering inside of y, W, U, V. By similar argument, the *second* event occurs with probability

$$\frac{1}{l+k+r+2} \cdot \frac{1}{l+k+1}$$

Summing over the two cases, we have

$$\begin{aligned} &\Pr[y < N(y) \wedge x < N(x)] \\ &= \frac{1}{l+k+r+2} \cdot \frac{1}{k+r+1} + \frac{1}{l+k+r+2} \cdot \frac{1}{l+k+1} \\ &= \frac{l+2k+r+2}{(l+k+r+2)(k+r+1)(l+k+1)} \end{aligned}$$

Finally, the conditional probability can be calculated as:

$$\begin{aligned} \Pr[y < N(y) \mid x < N(x)] &= \frac{\Pr[y < N(y) \wedge x < N(x)]}{\Pr[x < N(x)]} \\ &= \frac{(l+r+2k+2)}{(l+k+r+2)(k+r+1)} \end{aligned} \qquad (1)$$

◀

B.2.1 The ε-min-wise property:

In Lemma 7, we assume uniform random permutation, but the same claim still holds with ε-min-wise property, since we only consider the probability of an element being the smallest. The proof of ε-min-wise property has similar structure with the proof of Lemma 7, except we need to take $1 \pm \varepsilon$ into account every time we calculate the probability of an element being smallest. We also observe that the statement "knowing x being the smallest across y, W, U, V does not reveal anything about the ordering inside of y, W, U, V" requires a small amount of extra independence in the $\mathcal{O}(\log \varepsilon^{-1})$-wise hash function. This has no effect on the asymptotic space bound for the hash function.

Therefore, we have $\Pr[x < N(x)] = (1 \pm \epsilon)/(l + k + 1)$ and

$$\begin{aligned}
&\Pr[y < N(y) \wedge x < N(x)] \\
&= \frac{1 \pm \varepsilon}{l + k + r + 2} \cdot \frac{1 \pm \varepsilon}{k + r + 1} + \frac{1 \pm \varepsilon}{l + k + r + 2} \cdot \frac{1 \pm \varepsilon}{l + k + 1} \\
&= (1 \pm \varepsilon)^2 \frac{l + 2k + r + 2}{(l + k + r + 2)(k + r + 1)(l + k + 1)}
\end{aligned}$$

And finally

$$\begin{aligned}
\Pr[y < N(y) \mid x < N(x)] &= \frac{\Pr[y < N(y) \wedge x < N(x)]}{\Pr[x < N(x)]} \\
&= (1 \pm \varepsilon) \frac{(l + r + 2k + 2)}{(l + k + r + 2)(k + r + 1)}
\end{aligned} \tag{2}$$

B.3 Proof of Lemma 8

Proof of Lemma 8. We persist with the binary random variables from the proof of Lemma 6. In the proof of Lemma 6, we show that $\Pr[X_v = 1] = \frac{(1 \pm \varepsilon)p}{d(v)+1}$. For simplicity, we ignore the $(1 \pm \varepsilon)$ factor in the following analysis, that is, $\Pr[X_v = 1] = \frac{p}{d(v)+1}$; the same analysis holds without removing $(1 \pm \varepsilon)$. For each X_v, we have

$$\mathrm{Var}(X_v) = E[X_v^2] - E^2[X_v] < \frac{p}{d(v) + 1}(1 - \frac{p}{d(v) + 1}) < \frac{p}{d(v) + 1}$$

For non-adjacent vertices i and j share $k \geq 0$ common neighbors, the covariance between X_i and X_j is

$$\begin{aligned}
&\mathrm{Cov}(X_i, X_j \mid (i, j) \notin E) \\
&= E[X_i X_j] - E[X_i]E[X_j] \\
&= \Pr[X_i = 1 \wedge X_j = 1] - \Pr[X_i = 1] \Pr[X_j = 1] \\
&= \Pr[X_i = 1 \mid X_j = 1] \Pr[X_j = 1] - \Pr[X_i = 1] \Pr[X_j = 1] \\
&= (\frac{p(d(i) + d(j) + 2)}{(d(i) + 1)(d(i) + d(j) - l + 2)} - \frac{p}{d(i) + 1}) \frac{p}{d(j) + 1} \\
&= \frac{p^2 l}{(d(i) + d(j) - l + 2)(d(i) + 1)(d(j) + 1)},
\end{aligned} \tag{3}$$

where the second-last equality arises from Equation (1), where we set $d(i) = l + k$ and $d(j) = k + r$. It is not hard to see that Equation (3) is a constraint function subject to $d(i) \geq 1$, $d(j) \geq 1$, and $k \leq \min\{d(i), d(j)\}$. This function has global maximum at $\frac{p^2 k}{(k+1)^2(k+2)}$ when $k = d(i) = d(j)$ for different values of $d(i)$ and $d(j)$, which is maximized at $\frac{p^2}{12}$ when $k = 1$. As indicated in Equation (2), when considering ε-min-wise property instead of uniform random permutation, the global maximum is $\frac{(1\pm\epsilon)p^2}{12} \leq \frac{p^2}{8}$ if $\varepsilon < 1/2$.

And for adjacent vertices i and j, X_i and X_j are strongly negatively correlated, as $X_i = 1$ implies $X_j = 0$. Hence,

$$\mathrm{Cov}(X_i, X_j \mid (i,j) \in E) = E[X_i X_j] - E[X_i]E[X_j] < 0$$

Let P be the number of vertex pairs that share at least one common neighbor. Assume the vertices are labeled from 1 to n, by variance equation for the sum of correlated variables, we have (for simplicity, we sometimes abbreviate $1 \leq i < j \leq n$ as $i < j$),

$$\begin{aligned} \mathrm{Var}(X) &= \sum_{i=1}^{n} \mathrm{Var}(X_i) + 2\sum_{i<j} \mathrm{Cov}(X_i, X_j) \\ &< \sum_{i=1}^{n} \mathrm{Var}(X_i) + 2\sum_{i<j \wedge (i,j)\notin E} \mathrm{Cov}(X_i, X_j) \\ &< \sum_{i=1}^{n} \frac{p}{d(i)+1} + 2\sum_{i<j \wedge (i,j)\notin E} \frac{p^2 k}{(k+1)^2(k+2)} \\ &\leq p\,\lambda + 2p^2 \frac{P}{8} \end{aligned} \tag{4}$$

Note that each vertex v introduces at most $\frac{d_v(d_v-1)}{2} \leq \frac{d_v^2}{2}$ new pairs of vertices that share a common neighbor, i.e., v. Thus, $P \leq \frac{1}{2}\sum_{v\in V(G)} d_v^2$. Since $\Delta \leq \frac{\varepsilon^2 n}{3(\bar{d}+1)^3}$, we have

$$\begin{aligned} \sum_{v\in V(G)} d_v^2 \leq \Delta^2 \frac{\bar{d}n}{\Delta} + (n - \frac{\bar{d}n}{\Delta}) &< (\Delta\bar{d} + 1)n \\ &\leq (\frac{\varepsilon^2 n\bar{d}}{3(\bar{d}+1)^3} + 1)n \\ &< \frac{\varepsilon^2 n^2}{3(\bar{d}+1)^2} + n \\ &\leq \frac{\varepsilon^2 n^2}{2(\bar{d}+1)^2}\,, \end{aligned} \tag{5}$$

where the last inequality holds when $n \geq \frac{\sqrt{6}(\bar{d}+1)}{\varepsilon}$. And the first inequality is because given the maximum degree Δ and a sufficiently large n, the term $\sum_{v\in V(G)} d_v^2$ is maximized when the degree distribution is highly biased. That is, when there are around $\frac{\bar{d}n}{\Delta}$ vertices hitting the maximum degree Δ, while the rest of vertices are having degree 1 (as the graph is assumed to be connected).

Finally, combining Equation (4) and (5), we have

$$\begin{aligned}\mathrm{Var}(X) \le p\,\lambda + 2p^2\frac{P}{8} &\le p\,\lambda + p^2\frac{\varepsilon^2 n^2}{16(\bar{d}+1)^2}\\ &\le p\,\lambda + p^2\frac{\varepsilon^2\lambda^2}{16}\\ &= E[X] + \frac{\varepsilon^2 E^2[X]}{16}\\ &\le \frac{\varepsilon^2 E^2[X]}{3}\end{aligned}$$

where the third inequality holds because the Turán Bound is at most the Caro-Wei Bound, i.e., $\frac{n}{\bar{d}+1} \le \sum_{v\in V(G)} \frac{1}{d(v)+1} = \lambda$ (see [4] for a proof). The inequality in the second-last line arises from $E[X] = p\lambda$. And the last inequality holds if $E[X] \ge \frac{16}{7\varepsilon^2}$, since we have assumed that $p \ge \frac{4(\bar{d}+1)}{\varepsilon^2 n}$ and $\lambda \ge \frac{n}{\bar{d}+1}$, $E[X] = p\lambda \ge \frac{4}{\varepsilon^2}$, thus the inequality follows.

◀

B.4 Proof of Theorem 5

Proof of Theorem 5. From Lemma 6 and Lemma 8, the expectation of the returned result, $\widehat{\lambda}$, is $(1\pm\varepsilon)\lambda$, and $\mathrm{Var}(X) \le \frac{\varepsilon^2 E^2[X]}{3}$. Hence, by Chebyshev's inequality,

$$\Pr\left[|\widehat{\lambda}-\lambda| \ge 3\varepsilon\lambda\right] \le \Pr\left[|\widehat{\lambda}-E[\widehat{\lambda}]| \ge \varepsilon E[\widehat{\lambda}]\right] = \Pr\left[|X-E[X]| \ge \varepsilon E[X]\right] \le \frac{1}{3},$$

where the first inequality holds because $\varepsilon < 1$ so that $(1+\varepsilon)^2 < (1+3\varepsilon)$ and $(1-\varepsilon)^2 > (1-3\varepsilon)$. Therefore, the algorithm succeeds with probability at least $2/3$.

The algorithm stores an ε-min-wise hash function and $\mathcal{O}(\bar{d}\varepsilon^{-2})$ vertices. The hash function takes $\mathcal{O}(\log\varepsilon^{-1}\cdot\log n)$ bits to store, and each vertex v takes $\mathcal{O}(\log n)$ bits to store. Therefore, the total space usage is $\mathcal{O}((\bar{d}\varepsilon^{-2}+\log\varepsilon^{-1})\log n) = \mathcal{O}(\bar{d}\varepsilon^{-2}\log n)$. ◀

B.5 Proof of Theorem 9

Proof of Theorem 9. Let a binary random variable X_v denote whether v is not removed by our algorithm. Since v is removed if and only if it does not have the smallest hash value across $N[v]$, $\Pr[X_v = 1] = \frac{1\pm\varepsilon}{d(v)+1}$. Let $X = \sum_{v\in V(G)} X_v$. By the construction of the algorithm, X is exactly the size of the final solution. And

$$E[X] = \sum_{v\in V(G)} \Pr[X_v = 1] = (1\pm\varepsilon)\sum_{v\in V(G)} \frac{1}{d(v)+1} = (1\pm\varepsilon)\,\lambda$$

Following the same proof as Lemma 8 (note: since we are not sampling, the sampling probability p in Lemma 8 can be regarded as 1), we can bound the variance of X as $\mathrm{Var}(X) \le \frac{\varepsilon^2 E^2[X]}{3}$. Applying Chebyshev's inequality, we have

$$\Pr\left[|\widehat{\lambda}-\lambda| \ge 3\varepsilon\lambda\right] \le \frac{1}{3},$$

Hence, the algorithm fails with probability at most $1/3$.

The algorithm only uses a randomly drawn ε-min-wise hash function, which can be stored using $\mathcal{O}(\log\varepsilon^{-1}\log n)$ bits and has calculation time $\mathcal{O}(\log\varepsilon^{-1})$. Hence, the space usage and the update time our algorithm is $\mathcal{O}(\log\varepsilon^{-1}\log n)$ and $\mathcal{O}(\log\varepsilon^{-1})$.

◀

The success probability can be boosted to $1-\delta$ by running $\log\delta^{-1}$ independent instances simultaneously and return the maximum result. By the union bound, the probability that all instances fail is at most $(\frac{1}{3})^{\log\delta^{-1}} = \delta$. After boosting, the new algorithm uses space $\mathcal{O}(\log\varepsilon^{-1}\log\delta^{-1}\log n)$ and has update time $\mathcal{O}(\log\varepsilon^{-1}\log\delta^{-1})$.

B.6 Proof of Lemma 10

Proof of Lemma 10. The upper bound, $\lambda(G) \geq \lambda(G_{-Deg_{\geq\mu}})$, clearly holds as removing vertices does not increase λ.

Given the average degree $\bar{d}$, by the Pigeonhole Principle, there are at most $\frac{\bar{d}n}{\mu}$ vertices with degree at least μ. Hence, we have

$$\begin{aligned}\lambda(G_{-Deg_{\geq\mu}}) > \lambda(G) - \sum_{v\in Deg_{\geq\mu}} \frac{1}{d(v)+1} &\geq \lambda(G) - \frac{1}{\mu+1}\frac{\bar{d}n}{\mu} \\ &> \lambda(G) - \varepsilon\frac{\bar{d}n}{(\bar{d}+1)^2} \\ &\geq \lambda(G) - \varepsilon\lambda(G),\end{aligned}$$

where the last inequality holds as the Turán Bound is at most the Caro-Wei Bound, i.e., $\frac{n}{\bar{d}+1} \leq \sum\frac{1}{d(v)+1} = \lambda$ (see [4] for a proof).

◀

B.7 Proof of Lemma 11

Proof of Lemma 11. By the construction of Subroutine 2, vertices in R are removed from the neighborhood list of sampled vertices (line 15 – 17). Hence, by the condition on line 20, each sampled vertex is retained in S if and only if it has the smallest hash value across its neighbors in the subgraph after removing R. Let X_v be the binary indicator of v being sampled and retained in S. Since the sampling procedure and the hash function are independent, $\Pr[X_v = 1] = (1\pm\varepsilon)\frac{p}{d'(v)+1}$, where $d'(v)$ is the degree of v after removing R. Let $X = \sum_{v\in V(G)} X_v$. Clearly, $X = |S|$ at the end of stream. Let λ' be the Caro-Wei bound of the subgraph after removing R, by the linearity expectation, the expectation of $\widehat{\lambda}'$ is

$$E[\widehat{\lambda}'] = E[|S|]/p = \frac{1}{p}\sum_{v\in V(G)} E[X_v] = \sum_{v\in V(G)} \frac{1\pm\varepsilon}{d'(v)+1} = (1\pm\varepsilon)\lambda'\,,$$

The variance of $\widehat{\lambda}'$ is the same as the variance of X, as p is fixed. By Lemma 8, condition on R contains all vertices with degree at least $\frac{\varepsilon^2}{3}\frac{n}{(\bar{d}+1)^3}$, we have $\mathrm{Var}(X) \leq \frac{\varepsilon^2 E^2[X]}{3}$. Hence,

$$\Pr\left[|\widehat{\lambda}' - \lambda'| \geq 3\varepsilon\lambda'\right] \leq \Pr\left[|X - E[X]| \geq \varepsilon E[X]\right] \leq \frac{\mathrm{Var}(X)}{\varepsilon^2 E^2[X]} \leq \frac{1}{3}$$

By Lemma 10, condition on R contains no vertices with degree smaller than $\frac{\bar{d}(\bar{d}+1)}{\varepsilon}$, $\lambda' \geq (1-\varepsilon)\lambda$. Also, the algorithm aborts if the size of sampled vertices and their neighborhood lists are too large (line 11 – 12). Since the average degree is $\bar{d}$, in expectation, $\bar{d}pn$ vertices are stored by our algorithm. By Markov's inequality, the probability that the actual number of stored vertices is 10 times greater than its expectation, $\bar{d}pn$, is at most 1/10. Applying the union bound, the probability that this algorithm fails or aborts is at most 11/30.

The space usage of this algorithm is space to store the hash function, plus the space to store sampled vertices and their neighbors. Because of our constraints on line 11, there are at most $\mathcal{O}(\bar{d}pn) = \mathcal{O}(\bar{d}^2\varepsilon^{-2})$ vertices being stored, where each vertex takes $\mathcal{O}(\log n)$ bits to store. The hash function takes $\mathcal{O}(\log \varepsilon^{-1} \log n)$ bits to store. Hence, the total space usage is $\mathcal{O}(\bar{d}^2\varepsilon^{-2}\log n)$.

◀

B.8 Proof of Theorem 13

Proof of Theorem 13. Firstly, note that $\widehat{\lambda}$ is an unbiased estimator of λ, as each vertex arrives uniformly at random. Let binary random variable X_v denote whether c is incremented when v arrives. That is, $X_v = 1$ if the neighborhood list of v is empty, and the counter is increased. These two events are independent, the former event happens with probability $\frac{1}{d(v)+1}$, while the latter event happens with probability p. Let $X = \sum_{v\in V(G)} X_v$. By the linearity of expectation, we have

$$E[\widehat{\lambda}] = E[c]/p = E[X]/p = \frac{1}{p}\sum_{v\in V(G)} E[X_v] = \sum_{v\in V(G)} \frac{1}{d(v)+1} = \lambda$$

By Lemma 8, the variance of X, $\mathrm{Var}(X) \le \frac{\varepsilon^2 E^2[X]}{3}$. Applying Chebyshev's inequality, we have

$$\Pr\left[|\widehat{\lambda} - \lambda| \ge \varepsilon\lambda\right] = \Pr\left[|X - E[X]| \ge \varepsilon E[X]\right] \le \frac{\mathrm{Var}(X)}{\varepsilon^2 E^2[X]} \le \frac{1}{3},$$

The algorithm might fail if the counter, c, becomes too large, i.e., $c > 10pn$. By Markov's inequality,

$$\Pr[X \ge 10pn] \le \frac{E[X]}{10pn} = \frac{\lambda}{10n} \le \frac{1}{10}$$

Therefore, applying the union bound, the algorithm succeeds with probability at least $1 - 1/3 - 1/10 = 19/30$.

The space usage of the algorithm is the size of the counter c. By the construction of the algorithm, c is incremented to at most $10pn$. Hence, it can be stored in $\mathcal{O}(\log(pn)) = \mathcal{O}(\log(\bar{d}\varepsilon^{-2}))$ bits.

◀

Similarly, by running $\log\delta^{-1}$ independent instances concurrently and return the maximum result, the success probability can be boosted to $1-\delta$. This is because the probability that all instances fail is at most $(\frac{11}{30})^{\log\delta^{-1}} = \delta$. The space usage of the new algorithm after boosting is $\mathcal{O}(\log(\bar{d}\varepsilon^{-2})\log\delta^{-1})$.

C Omitted Proofs from Section 4

C.1 Proof of Theorem 17

To start with, note that

▶ **Lemma 36.** *In every connected graph with $n \geq 3$, there is a maximum independent set containing all leaves and no support vertices.*

Proof of Lemma 36. This can be proved via adjusting vertices from a given maximum independent set. Given a maximum independent set, if it contains all leaves, then we are done. If some leaves are not included, then their support vertices must be in the independent set. This can be seen via contradiction, assume that both the support vertex and its leaves are not in the independent set, then clearly we could add all the leaves in the set, which violates the assumption that the set is maximum. Note that each support vertex must be adjacent to at least one leaf, and if it is in the independent set, none of its leaves is in the set. Hence, we could remove all support vertices from the given independent set, and add all their leaves into the set, the size of the resulting set does not decrease. Given that the original set is already maximum, the new set is also maximum. ◀

Hence, in the proof of Theorem 17, we assume that the max independent set of every tree contains every leaf unless $n = 2$.

Proof of Theorem 17. To prove Theorem 17, it suffices to show that, for every tree T with $|T| \geq 2$,

$$\max\left\{\frac{|T|}{2}, |Deg_1(T)| - 1\right\} \leq \beta(T) \leq \frac{|T| + |Deg_1(T)|}{2}$$

First we show the lower bound for $\beta(T)$. Since a tree is bipartite and each side of the bipartition forms an independent set, it follows that $\beta(T) \geq |T|/2$. Also, when $|T| = 2$, $|Deg_1(T)| = 2$ and $\beta(T) = 1$. When $|T| \geq 3$, no two leaves can be adjacent to each other and hence the set of all leaves forms an independent set, i.e., $\beta(T) \geq |Deg_1(T)|$. Summarizing, we have $\beta \geq \max\left\{\frac{|T|}{2}, |Deg_1(T)| - 1\right\}$.

Next, we show that $\beta(T) \leq \frac{|T|+|Deg_1(T)|}{2}$ by induction on $|T|$. We say that a pair of leaves of a tree are twins if they share a common neighbor. We have two cases to consider depending on whether T has twins or not:

Case 1: T has twins Let the twins be v, w and their shared neighbor be u. Delete w, and let the tree obtained be $T' = T - w$. Then we have $|T'| = |T| - 1$, $|Deg_1(T')| = |Deg_1(T)| - 1$, and $\beta(T') = \beta(T) - 1$. By induction hypothesis, we know that

$$\begin{aligned}
&|T'| + |Deg_1(T')| \geq 2\beta(T') \\
&\Rightarrow |T| - 1) + \big(|Deg_1(T)| - 1\big) \geq 2\big(\beta(T) - 1\big) \\
&\Rightarrow |T| + |Deg_1(T)| \geq 2\beta(T)
\end{aligned}$$

Case 2: T does not have twins: Let a leaf w have a parent u whose parent is x. We can assume that u has no other leaves adjacent to it since T has no twins. Delete both u and w, and let the obtained tree be $T' = T - \{u, w\}$. It is easy to see that $|T'| = |T| - 2$ and $|Deg_1(T)| \geq |Deg_1(T')| \geq |Deg_1(T)| - 1$, because x may or may not be a leaf in T' but w was definitely a leaf in T. Given an independent set I in T, we can obtain an independent set $I' = I - w$ in T'. Similarly, given an independent set H' in T', we can obtain an independent set $H = H' \cup \{w\}$ in T. Hence, it follows that $\beta(T') = \beta(T) - 1$. By induction hypothesis, we have

$$\begin{aligned}
&|T'| + |Deg_1(T')| \geq 2\beta(T') \\
&\Rightarrow \big(|T| - 2\big) + (|Deg_1(T')|) \geq 2(\beta(T) - 1) \\
&\Rightarrow |T| + |Deg_1(T)| \geq 2\beta(T).
\end{aligned}$$

◀

C.2 Proof of Theorem 18

By Lemma 36, we will assume that (unless $n = 2$) we assume that the maximum independent set contains every leaf and none of the support vertices. We first show a lower bound (Claim 37) and upper bound (Claim 38) on the independence number of trees. Theorem 18 then follows immediately by applying Claim 37 and Claim 38 to each tree component of the forest F.

▷ Claim 37. For every tree T with $|T| \geq 2$,

$$\beta \geq \frac{1}{2}(|T| + |Deg_1(T)| - |Supp(T)|)$$

Proof. We prove the claim by showing that there exists an independent set of size at least $\frac{1}{2}\big(|T| + |Deg_1(T)| - |Supp(T)|\big)$. When $|T| = 2$, then T is a path on two vertices which implies $\beta(T) = 1$ and $|Deg_1| = 2 = |S| = 2$ and the desired inequality holds. Hence, we assume $n > 2$.

It suffices to show that there exists an independent set H containing all leaves and half of the vertices from $Deg_{\geq 2}(T)$ vertices that are not support vertices. We first remove all leaves and support vertices from T to obtain a forest F' with $|Deg_{\geq 2}(T)| - |Supp(T)|$ vertices. Since each forest is bipartite and each side of a bipartition forms an independent set, it follows that F' has an independent set H' of size $\geq |F'|/2$. We take H to be the union of $Deg_1(T)$ and H': it follows that H is an independent set in T since F' does not contain any support vertices of T. The size of H is $|Deg_1(T)| + \frac{|Deg_{\geq 2}(T) - Supp(T)|}{2} = \frac{|T| + |Deg_1(T)| - Supp(T)}{2}$ since $|T| = |Deg_1(T)| + |Deg_{\geq 2}(T)|$. This concludes the proof of Claim 37. ◀

The lower bound of Claim 37 is tight: consider a path P on $2n$ vertices. Then $\beta(P) = n$ and $|Deg_1(P)| = 2 = |Supp(P)|$.

▷ Claim 38. For every tree T with $|T| \geq 2$,

$$\beta(T) \leq \frac{2}{3}(|T| + |Deg_1(T)| - |Supp(T)|)$$

Proof. We prove this lemma via induction on the graph order, n. Consider all trees with $2 \leq n \leq 4$ as the base cases: these can all be easily verified by hand. Assume the inequality holds for every tree with order $n - 1$. We say that two leaves are twins if they share a common parent. There are two cases to consider depending on whether T has twins or not:

Case 1: There exist twins in T. Let T' be the tree after removing either one of the twins. Then we have $|T| = |T'| + 1$, $|Deg_1(T)| = |Deg_1(T')| + 1$, and $|Supp(T)| = |Supp(T')|$. According to Lemma 36, we have that $\beta(T) = \beta(T') + 1$. Hence, by the induction hypothesis, the inequality holds.

Case 2: There are no twins in T. Consider the longest path in T, denote one of its leaves as x, the parent of x as y, the parent of y as w, and the parent of w as z. If T has more than four nodes and no twins, then z is guaranteed to be non-leaf. This can be seen by contradiction. Assuming z is a leaf and there is more than four nodes. By our choice of the longest path, the neighbors of w and y can only be leaves. Since there is no twin, w and y both have degree two, there are exactly four nodes, a contradiction. By similar argument, it is not hard to see that y must have degree two.

Case 2.1: $\mathbf{d(w) > 2}$. Remove both x and y, and let the resulting tree be T'. Note that this removes a leaf and a support vertex. Because w has degree more than 2, the removal of x and y does not make w a leaf, and w is a support vertex in T' if and only if it is a support vertex in T. Therefore, we have $|T| = |T'| + 2$, $|Deg_1(T)| = |Deg_1(T')| + 1$, and $|Supp(T)| = |Supp(T')| + 1$. Moreover, we claim that $\beta(T) = \beta(T') + 1$. This is because by Lemma 36, there is a max independent set H such that $x \in H$ and $y \notin H$: thus the existence of x and y has no impact on whether w and the rest of vertices are in $\beta(T)$ or not, i.e., they are in $\beta(T)$ if and only if they are also in $\beta(T')$. By the induction hypothesis, the inequality holds.

Case 2.2: $\mathbf{d(w) = 2}$. We have two cases to consider.

Case 2.2.1: $\mathbf{z} \notin \mathit{Supp}(\mathbf{T})$. We remove x and y. Similar to Case 2.1, we have $|T| = |T'|+2$ and $\beta(T) = \beta(T') + 1$. However, since $d(w) = 2$ and z is not a support vertex in T, the removal of x and y makes w a leaf and z a support vertex. Hence $|Deg_1(T)| = |Deg_1(T')|$, and $|Supp(T)| = |Supp(T')|$. By the induction hypothesis, the inequality holds.

Case 2.2.2: $\mathbf{z} \in \mathit{Supp}(\mathbf{T})$. Remove x, y, and w from T and denote the remaining graph as T'. Since z is a support vertex, according to Lemma 36, there is a max indpendent set H of T such that $z \notin H$ which implies that w must in the maximum independent set. Therefore, we have $|T| = |T'| + 3$ and $\beta(T) = \beta(T') + 2$. Moreover, removing these vertices make z neither a leaf nor a support vertex in T', unless $d(z) = 2$. In the former case, we have $|Deg_1(T)| = |Deg_1(T')| + 1$, and $|Supp(T)| = |Supp(T')| + 1$. By the induction hypothesis, the inequality holds. In the latter case, T is a path of length 5: the claim can easily verified by hand for this specific graph. This concludes the proof of Claim 38. ◀

This upper bound of Claim 38 is asymptotically tight: consider the graph constructed from a star on $(r + 1)$ vertices by replace each of the r leaves with a path consisting of 3 vertices. Clearly, this graph is a tree, say T, with $3r + 1$ vertices, and $|Supp(T)| = |Deg_1(T)| = r$. Also, $\beta(T) = 2r$ as all the r leaves and their r grandparents form a maximum independent set.

C.3 Proofs of Theorem 19

Before showing our bounds, we first prove the following lemma.

▶ **Lemma 39.** *In every connected graph with $n \geq 3$, there is a minimum dominating set that contains all support vertices and no leaves.*

Proof. We prove this by showing that every minimum dominating set can be adjusted to contain no leaves and all support vertices without increasing its size. Clearly, in every connected graph with $n \geq 3$, all support vertices have degree greater than 1 (i.e., are not themselves leaves). If a support vertex has degree 1, then its neighbor must be a leaf, hence $n = 2$, violating our assumption about $n \geq 3$. Given a minimum dominating set, If it contains no leaves and all support vertices, then we are done. If not, we can remove the leaves from the dominating set and add their neighbors into the set. By the definition of support vertices, the neighbors of leaves are themselves support vertices. The resulting set is still a dominating set as the newly added support vertices dominate the removed leaves and themselves. Moreover, before the adjustment, for each pair of support vertex and leaf, at least one of them is in the dominating set, otherwise the domination property is broke. The adjustment ensures that the new dominating set contains all support vertices and no leaves. The size of the new dominating set is at most the size of original dominating set, because each removed leaf introduces at most one neighbor (i.e., support vertex) into the new set. ◀

Hence, in the following section, we assume the minimum dominating set contains all support vertices and no leaves. Now we give our upper and lower bounds on the domination number of forest. For the upper bound, we have

▶ **Lemma 40.** *For every forest F, we have*

$$\frac{|Deg_{\geq 2}(F)| + |Supp(F)|}{2} \geq \gamma(F)$$

Proof. If $|F| = 2$, then F has to be an edge. Every vertex of F is both a support vertex and a leaf, and we can pick either of them as the dominating set. Hence, $\frac{|Deg_{\geq 2}(F)|+|Supp(F)|}{2} = \frac{1+1}{2} = 1 = \gamma(F)$, the inequality holds.

Next, we prove the inequality for graphs with $n > 2$. It is known that if a graph G is split into k disjoint subgraphs, $G_1, G_2, \ldots, G_k$, we have $\gamma(G) \leq \gamma(G_1) + \gamma(G_2) + \cdots + \gamma(G_k)$. This is because connecting two or more disjoint graphs only introduces new edges, thus vertices do not become undominated. Rewriting the inequality as $|Supp(F)| + \frac{|Deg_{\geq 2}(F)|-|Supp(F)|}{2}$, the first term could be viewed as adding all vertices in $Supp(F)$ into the dominating set. By doing so, all of the vertices in $N\big[Supp(F)\big]$ are dominated (note that this includes all the leaves). The second term could be viewed as an upper bound on the domination number after removing all the leaves and support vertices. Clearly, the remaining graph is a forest with $(|Deg_{\geq 2}(F)| - |Supp(F)|)$ vertices. Thus it remains to prove that in such a forest, there exists a dominating set of size at most $\frac{|Deg_{\geq 2}(F)|-|Supp(F)|}{2}$.

Note that all isolated vertices in the induced graph are already dominated by some vertices in S, because a vertex becomes isolated only if all of its neighbors are in S. Thus, we only need to show that for each tree components (of size $k > 1$) in the remaining forest, there exists a dominating set of size at most $\frac{k}{2}$. Ore [39] proved that for every graph G with order n and no isolated vertices, $\gamma(G) \leq \frac{n}{2}$. Hence, this lemma follows. ◀

The upper bound of Lemma 40 is tight: construct a tree T by taking a path of on r vertices for any $r \geq 2$, add a leaf to every vertex on the path. It is easy to see that $|Supp(T)| = |Deg_{\geq 2}(T)| = \gamma(T) = r$.

Next, we present our lower bound result.

▶ **Lemma 41.** *For every tree T with $T| \geq 2$,*

$$\gamma(T) \geq \frac{|Deg_{\geq 2}(T)| + |Supp(T)|}{4}$$

Proof. We prove the lemma by induction on $n = |T|$. Consider the base cases as trees with $n \leq 4$, which can be easily verified by hand.

It remains to show the inductive step. Two or more leaves are twins if they share a common parent node (i.e., support vertex). If the tree contains twins, we remove any one of them from T to obtain a tree T'. By Lemma 39, the minimum dominating set of T does not contain either of the twins. Hence, we have that $\gamma(T) = \gamma(T')$, $|Deg_{\geq 2}(T)| = |Deg_{\geq 2}(T')|$, and $|Supp(T) = Supp(T')|$ stay the same. By induction hypothesis, Lemma 41 holds.

Next, assume there are no twins and consider the longest path, say P, in T. Denote the two endpoints as x and x' respectively, let y be the parent of x, w be the parent of y, and z be the parent of w. Note that the length of P must be greater than 4, thus z and x' must be different. This can be seen via contradiction. Assuming P has length 4, by our choice of longest path P, x and z (or equivalently, x') must be leaves, and the neighbors of y and w must be also leaves. Otherwise, our choice of the longest path is violated. If y or w is adjacent to leaves other than x and z, then there exist twins in the graph, violating our assumption of no twins. Hence, the graph contains exactly 4 vertices, x, y, w, and z. It should be considered as the base case. Similar arguments can be obtained for P with length less than 4. Hence, P's length must be greater than 4.

If w is a support vertex, then we are done by induction. By Lemma 39, the minimum dominating set contains both w and y as they are support vertices. Note that y must have degree exactly 2, otherwise the graph has either a twin, or a path longer than P. Delete x and let the tree obtained be T''. Then y becomes a leaf which is already dominated by w. Therefore, after removing x, we have $\gamma(T'') = \gamma(T) - 1$. Since y becomes a leaf, we have $|Deg_{\geq 2}(T'')| = |Deg_{\geq 2}(T)| - 1$ and $|Supp(T'')| = |Supp(T)| - 1$, and hence the inequality holds by our induction hypothesis.

It remains to show that $\gamma(T) \geq \frac{|Deg_{\geq 2}(T)|+|Supp(T)|}{4}$ when there are no twins and w is not a support vertex. Let T' and T'' be the two sub-trees formed by deleting the edge (w, z), such that T' contains w and T'' contains z. Note that by our previous analysis, only paths of length 2 can be incident on w, otherwise the tree falls into previous cases. Denote the number of such paths as r; clearly, $r \geq 1$ because there is path from x to w. In tree T', clearly, by picking all r child nodes of w, we obtain a minimum dominating set. Since w is not included in the set, $\gamma(T) = \gamma(T') + \gamma(T'') = r + \gamma(T'')$. Let $\psi(T) = |Deg_{\geq 2}(T)| + |Supp(T)|$. We have the following three inequalities:

1. if $d(z) > 1$ in T'': $Deg_{\geq 2}(T) \leq Deg_{\geq 2}(T'') + (r + 1)$
2. if $d(z) = 1$ in T'': $Deg_{\geq 2}(T) \leq Deg_{\geq 2}(T'') + (r + 1) + 1$
3. $Supp(T) \leq Supp(T'') + r$

The first two inequalities hold because T' contains r vertices from $Deg_{\geq 2}(T')$ vertices, and by adding the edge (w, z) we have $d(w) \geq 2$. Also, if z has degree 1 in T'', z is not in $Deg_{\geq 2}(T'')$, but adding the edge (w, z) puts it in $Deg_{\geq 2}(T)$. Otherwise, z is already in $Deg_{\geq 2}(T'')$. Similarly, the *third* inequality holds because there are r vertices in $Supp(T')$, adding the edge (w, z) does not introduce new support vertices in $Supp(T)$. Hence, $Supp(T) \leq Supp(T'') + r$. This is an inequality because adding the edge (w, z) makes z non-leaf, thus a support vertex might be removed. Therefore, we have $\psi(T) \leq \psi(T'') + 2r + 2$. And:

$$\begin{aligned}
\gamma(T) &= \gamma(T'') + r \\
&\geq \frac{|Deg_{\geq 2}(T'')| + |Supp(T'')|}{4} + r \qquad \text{[by induction hypothesis]} \\
&= \frac{|Deg_{\geq 2}(T'')| + |Supp(T'')| + 4r}{4} \\
&\geq \frac{|Deg_{\geq 2}(T)| + |Supp(T)|}{4}
\end{aligned}$$

where the last inequality holds because $4r \geq 2r + 2$ when $r \geq 1$, which is always true since there is a path from w to x.

◀

The lower bound in Lemma 41 is asymptotically tight. Consider the tree T containing a vertex v with one leaf and r paths of length 4 (P_4) incident on it, which can be also viewed as a star graph with $r + 1$ leaves except r of them are replaced with P_4. Clearly, we have $\gamma(T) = r + 1$. Meanwhile, we have $|Deg_{\geq 2}(T)| = 3r + 1$, and $|Supp(T)| = r + 1$. Hence,

$$\frac{|Deg_{\geq 2}(T)| + |Supp(T)|}{4} = \frac{4r + 2}{4} \approx r + \frac{1}{2}.$$

When $r \to \infty$, $\frac{r+1}{r+\frac{1}{2}} \to 1$.

Combining Lemma 40 and 41, we have Theorem 19.

C.4 Proofs of Theorem 20

Before establishing our upper bound and lower bound on ϕ, we prove the following lemma.

▶ **Lemma 42.** *In every connected graph with $n \geq 3$, there is a maximum matching such that every support vertex is matched by one of its edges between it and its leaves.*

Proof. The first half of the lemma (every support vertex is matched) can be proved via contradiction. Assume we have a maximum matching and a support vertex s is not matched, then we could include an edge between s and one of its leaves into the matching, a contradiction.

The latter half can be proved via alternating edges in every given maximum matching without decreasing its size. Given a maximum matching M. If M satisfies the condition, then we are done. If there is a support vertex s that is matched through an edge between it and a non-leaf vertex, then by the definition of matching, none of its leaves are matched. Hence, we can remove the matching edge of s and include one of the edges between s and its leaves into the matching. This is valid as the leaf is not matched before, and s is matched only by the newly added edge. The size of maximum matching does not decrease.

◀

We may now prove an upper bound on ϕ,

▶ **Lemma 43.** *For every tree T with $n \geq 2$,*

$$\phi \leq \frac{|Deg_{\geq 2}(T)| + |Supp(T)|}{2}$$

Proof. We prove the lemma by induction on $n = |T|$. If $|T| = 2$, then T is an edge and we have $\phi(T) = 1, |Deg_{\geq 2}(T)| = 0$ and $|S| = 2$: hence the inequality holds. Henceforth, we assume that $n > 2$.

Rearranging the upper bound, we have

$$|Supp(T)| + \frac{|Deg_{\geq 2}(T)| - |Supp(T)|}{2}\,.$$

By Lemma 42, all support vertices are matched by the edges between them and one of their leaves. Hence, removing all support vertices and their leaves removes $|S|$ edges in the matching. The remaining graph is a forest F with $n(F) = |Deg_{\geq 2}(T)| - |S|$, and all edges in F can be added into the matching as they do not share a common endpoint with edges between support vertices and leaves (i.e., the matching edges that are removed). Hence it remains to prove that for every forest, there is a matching of size at most $\frac{n(F)}{2}$. Since every forest is bipartite, and in a bipartite graph with total order of n, the matching number is at most $\frac{n}{2}$. This is because one side has at most $\frac{n}{2}$ nodes. By the Pigeonhole Principle, if there is a matching of size greater than $\frac{n}{2}$, at least one node has more than one matching edges incident on it, a contradiction.

◀

The upper bound of Lemma 43 is tight: consider the tree T obtained by taking a star graph H with r leaves and attaching a new leaf for each vertex in H (including the center). It is easy to see that $|Deg_{\geq 2}(T)| = |Supp(T)| = r + 1$, as all vertices in H become a support vertex and have more than 1 neighbor. And it is not hard to see that $\phi = r + 1$, as the maximum matching is the edge set $E(T) \setminus E(H)$.

The next lemma gives a lower bound for the matching number:

▶ **Lemma 44.** *For every tree T with $|T| \geq 2$,*

$$\phi(T) \geq \frac{|Deg_{\geq 2}(T)| + |Supp(T)|}{3}$$

Proof. We prove the lemma by induction on $n = |T|$. We consider trees with $2 \leq n \leq 4$ as the base cases, which can be easily verified by hand. We say that a pair of vertices are twins if and only if they are leaves and adjacent to a common (support) vertex. There are two cases to consider depending on whether not T contains twins.

Case 1: There exist twins in T: Let x, y be the leaves which are twins and let s be the common vertex that they are adjacent to. Remove either of x or y, and let T' be the tree after the removal. Note that in T', s is a leaf if and only if $d(s) = 2$ in T, in this case $n = 3$, which falls into our base case. Hence, we have $|Deg_{\geq 2}(T)| = Deg_{\geq 2}(T')$ and $|Supp(T)| = |Supp(T')|$. Moreover, we claim that $\phi(T) = \phi(T')$: this is because only one of s's edges can be included into matching, if the removed leaf is on this edge, then we could add another edge between s and the other one of twin. By the induction hypothesis, this inequality holds.

Case 2: There are no twins in T: Consider the longest path in T, denote one of its endpoint as x, the parent of x as y, the parent of y as w, and the parent of w as z. By the induction hypothesis, T has more than 4 nodes and no twins, hence z is guaranteed to be a non-leaf. This can be seen by contradiction. Assuming z is leaf, by our choice of longest path, the neighbors of w and y can only be leaves. Otherwise, there exists a longer path. Also, since there is no twin, w and y both have degree 2, thus there is exactly 4 nodes, a contradiction. By similar argument, it is not hard to see that y is guaranteed to have degree 2.

Case 2.1: $\mathbf{d(w) > 2}$ We remove both x and y, denote the resulting tree as T'. Since $d(w) > 2$, removing x and y does not make w leaf, thus $|Deg_{\geq 2}(T)| = |Deg_{\geq 2}(T')|+1$. Also, w is a support vertex in T' if and only if it is a support vertex in T, we have $|Supp(T)| = |Supp(T')|+1$. By Lemma 42, we know that $e(x, y)$ is in the maximum matching of T, but not $e(y, w)$. Hence, the removal of x and y does not change the matching status of vertices in T': this implies that $\phi(T) = \phi(T') + 1$. By the induction hypothesis, the inequality holds.

Case 2.2: $\mathbf{d(w) = 2}$ We have two cases to consider:

Case 2.2.1: $\mathbf{z} \notin Supp(\mathbf{T})$ We remove x and y. Similar to Case 2.1, we have $\phi(T) = \phi(T') + 1$. However, since $d_w = 2$ and z is not a support vertex in T, the removal of x and y makes w a leaf and z a support vertex. Hence $|Deg_{\geq 2}(T)| = |Deg_{\geq 2}(T')| + 2$, and $|Supp(T)| = |Supp(T')|$. By the induction hypothesis, the inequality holds.

Case 2.2.2: $\mathbf{z} \in Supp(\mathbf{T})$ Consider T' as the tree after removing x, y, and w from T. Since z is a support vertex, according to Lemma 42, z must be matched and $e(x, y)$ is in the maximum matching. Hence, we know that $e(y, w)$ and $e(w, z)$ are not in the matching, and the removal of x, y, and w does not change the matching status of vertices in T'. Therefore, $\phi(T) = \phi(T') + 1$. Moreover, removing these three vertices does not make z a leaf or make it not a support vertex in T', unless z itself has degree 2. In the former case, we have $|Deg_{\geq 2}(T)| = |Deg_{\geq 2}(T')| + 2$, and $|Supp(T)| = |Supp(T')| + 1$. And by the induction hypothesis, the inequality holds. In the later case, we have a path on 5 veryices since x is a leaf, y, w, and z have degree 2, and z is also a support vertex. It can be easily verified by hand that the inequality holds for $P5$.

◀

The lower bound is asymptotically tight. Consider a graph G constructed as follows. Start with a star graph, H, with r leaves, and replace each leaf with a path of length 3. G has $2r + 1$ non-leaf vertices, including the star center and two of the vertices on each of the paths. Also $|Supp(G)| = r$ as only the middle vertices on the paths are support vertices. Lastly, a maximum matching can be found by adding all edges between support vertices and leaves, and one edge from the edges incident on star center, thus $\phi(G) = r + 1$.

Combining to Lemma 43 and Lemma 44, Theorem 20 follows.

C.5 Proof of Corollary 21

By definition, $n = |F| = |Deg_{\geq 2}(F)| + |Deg_1(F)|$ and $|Supp(F)| \leq |Deg_1(F)|$. Hence, if $|Supp(F)| \leq \frac{|Deg_{\geq 2}(F)|}{2}$, we have the following inequality,

$$|Supp(F)| \leq \frac{|Deg_{\geq 2}(F)|}{4} + \frac{|Deg_1(F)|}{2} = \frac{n + |Deg_1|}{4} \tag{6}$$

Utilizing Equation (6) and combining Theorem 17 and Theorem 18, if $|Supp(F)| \leq \frac{|Deg_{\geq 2}(F)|}{2}$, then

$$\frac{3(n + |Deg_1(F)|)}{8} \leq \frac{n + |Deg_1(F)| - |Supp(F)|}{2} \leq \beta(F)$$

By Theorem 17, we have $\beta(F) \leq \frac{n+|Deg_1(F)|}{2}$, Corollary 21 follows.

C.6 Proof of Corollary 22

Combining Theorem 19 and Theorem 16, if $|Supp(F)| \leq \frac{|Deg_{\geq 2}(F)|}{3}$, we have

$$\frac{|Deg_{\geq 2}(F)|}{3} \leq \gamma(F) \leq \frac{|Deg_{\geq 2}(F)| + |Supp(F)|}{2} \leq \frac{2|Deg_{\geq 2}(F)|}{3}$$

Hence, Corollary 22 follows.

C.7 Proof of Corollary 23

Combining Theorem 14 and Theorem 20, if $|Supp(F)| \leq \frac{|Deg_{\geq 2}(F)|}{3}$, we have

$$\phi(F) \leq \frac{|Deg_{\geq 2}(F)| + |Supp(F)|}{2} \leq \frac{3(|Deg_{\geq 2}(F)| + c(F))}{4}$$

Since by Theorem 14, $\phi(F) \geq \max\{c(F), \frac{|Deg_{\geq 2}(F)|+c(F)}{2}\}$, Corollary 23 follows.

C.8 Proof of Lemma 27

Proof of Lemma 27. We prove this lemma by showing the following claim

▷ Claim 45. For each tree T (with at least two vertices) we have $|Deg_1(T)| = 2 + \sum_{i=3}^{\Delta}(i - 2) \cdot |Deg_i(T)|$.

Proof.

$$\begin{aligned}
\sum_{i=3}^{\Delta}(i-2) \cdot |Deg_i(T)| &= \sum_{i=2}^{\Delta}(i-2) \cdot |Deg_i(T)| \\
&= \sum_{i=2}^{\Delta} i \cdot |Deg_i(T)| - 2 \cdot \sum_{i=2}^{\Delta}|Deg_i(T)| \\
&= \Big((2|T| - 2) - Deg_1(T)\Big) - 2 \cdot \Big(|T| - Deg_1(T)\Big) \\
&= Deg_1(T) - 2
\end{aligned}$$

In the penultimate line, we have used the fact that $\sum_{v \in T} d(v) = 2|T| - 2$. ◀

The proof of Lemma 27 then follows from Claim 45 by summing up over all tree components of F.

◀

C.9 Proof of Lemma 30

Proof of Lemma 30. We claim that, conditional on the algorithm does not abort at line 13, the candidate set, C, obtained at line 16 contains only vertices from $Supp(F)$, and if a vertex in $Supp(F)$ is sampled, it is kept in C. This is not hard to see, by performing a second pass, we obtain the exact degree of all sampled vertices and their neighbors. So that a vertex is included in C if there is at least one leaf adjacent to it, which is exactly the definition of $Supp(F)$.

It remains to bound the output of this algorithm, conditional on the algorithm does not abort. For all $i \in V(G)$, let the binary random variable X_i indicate whether a vertex v_i is a support vertex and is sampled in I. Let $X = \sum_{i \in G} X_i$. Clearly, since vertices are sampled uniformly with probability $p = \frac{c_1}{\varepsilon_1^2 K_1}$, we have

$$\begin{aligned} E[X] &= \sum_{i \in G} E[X_i] = \sum_{i \in G} \Pr[v_i \in (Supp(F) \cap I)] \\ &= \sum_{i \in Supp(F)} \Pr[v_i \in I] = \frac{c_1 \, |Supp(F)|}{\varepsilon_1^2 K_1} \end{aligned}$$

Thus by the algorithm design, $E[|\widehat{S}|] = \frac{\varepsilon_1^2 K_1}{c_1} \times E[X] = |Supp(F)|$. Moreover, when $|Supp(F)| \geq K_1$, we have

$$\begin{aligned} \Pr\left[\left||\widehat{S}| - |Supp(F)|\right| \geq \varepsilon_1 |Supp(F)|\right] &= \Pr\left[|X - E[X]| \geq \varepsilon_1 E[X]\right] \\ &\leq 2e^{-\frac{\varepsilon_1^2 E[X]}{3}} \\ &\leq 2e^{-c_1/3}, \end{aligned}$$

where we apply the negatively correlated Chernoff bound (Theorem 35) in the second-last inequality, since the X_i's are negatively correlated. And the last inequality holds as $|Supp(F)| \geq K_1$.

Next, we show that when $|Supp(F)| < K_1$, conditional on the algorithm does not abort, the probability that $|\widehat{S}|$ is larger than $2K_1$ is small. Again, by the negatively correlated Chernoff bound, we have

$$\begin{aligned} \Pr\left[|\widehat{S}| > 2K_1\right] &= \Pr\left[X > \frac{2K_1 E[X]}{|Supp(F)|}\right] \\ &< \Pr\left[X > (1 + \frac{K_1}{|Supp(F)|}) E[X]\right] \\ &\leq e^{-\frac{K_1^2 E[X]}{3|Supp(F)|^2}} \\ &\leq e^{-\frac{c_1 K_1}{3|Supp(F)|\varepsilon^2}} \\ &\leq e^{-\frac{c_1}{3\varepsilon^2}}, \end{aligned}$$

where the first and the last inequalities hold because $|Supp(F)| < K_1$.

Lastly, we bound the space usage of the algorithm and the probability that the algorithm aborts at line 13. By the design of the algorithm, at most $\frac{2m}{n} \frac{c_1 n}{\varepsilon_1^2 K_1} e^{\frac{c_1}{3}}$ vertices can be stored from the first pass, otherwise it aborts at line 13. By our assumption, the number of total edge insertions is bounded by $\mathcal{O}(n)$. Thus, we have $\frac{2m}{n} \in \mathcal{O}(1)$. Therefore, since each vertex requires $\mathcal{O}(\log n)$ bits to store its identifier, the space usage of the first pass is $\widetilde{\mathcal{O}}(\frac{n}{\varepsilon_1^2 K_1})$. Moreover, in the second pass, each degree counter takes an additional $\mathcal{O}(\log n)$ space to store. Hence, the total space usage is still $\widetilde{\mathcal{O}}(\frac{n}{\varepsilon_1^2 K_1})$.

In this algorithm, we sampled $\frac{c_1 n}{\varepsilon_1^2 K_1}$ vertices in advance. And the average degree of the graph at any point of the stream can be calculated as $\frac{2m}{n}$. Hence, if we pick a vertex uniformly at random, the expected number of its neighbors is bounded by $\frac{2m}{n}$. Let Y_i be the number of neighbors of vertex i, $E[Y_i] = \frac{2m}{n}$. Let $Y = \sum_{v \in V(G)} Y_v$ We have

$$E[Y] = \sum_{v \in V(G)} E[Y_v] = \frac{2m}{n} \frac{c_1 n}{\varepsilon_1^2 K_1}$$

The algorithm aborts if more than $\frac{2m}{n}\frac{c_1 n}{\varepsilon_1^2 K_1}e^{\frac{c_1}{3}}$ vertices are retained. Since we are sampling each vertex uniformly at random, Y_v's are independent of each other. By Markov inequality, the probability that the actual size is $e^{\frac{c_1}{3}}$ times larger than the expected size is at most $e^{-\frac{c_1}{3}}$.

Applying a union bound with the abort probability and the concentration probabilities above, we can bound the fail probability of our algorithm as follows. When $|Supp(F)| \geq K_1$, Algorithm 3 outputs an $(1\pm\varepsilon)$-estimate of $|Supp(F)|$ and fails with probability at most $2e^{-\frac{c_1}{3}} + e^{-\frac{c_1}{3}} = 3e^{-\frac{c_1}{3}}$. Similarly, when $|Supp(F)| < K_1$, the estimate returned by Algorithm 3 is larger than $2K_1$ with probability at most $e^{-\frac{c_1}{3\varepsilon_1^2}} + e^{-\frac{c_1}{3}} < 2e^{-\frac{c_1}{3}}$ since $\varepsilon_1 < 1$.

◀

C.10 Proof of Lemma 31

Proof of Lemma 31. Note that in the first pass, only vertices in $Deg_{\geq 2}(F)$ are sampled. This is because line 6 decreases the frequency of all vertices by one, so leaves has 0 frequency and by the definition of k-sparse recovery, they are not recovered by the sparse recovery data structure. Moreover, by Theorem 3, when $|Deg_{\geq 2}(F)| \leq K_2$ (i.e., when it is small), the sparse recovery fails with probability no more than $1/c_2$.

Since the sparse recovery data structure might output a false positive result. That is, a k-sparse vector when the original degree vector is not k-sparse. We need to perform a sanity check at line 20 to ensure that the returned result contains all $Deg_{\geq 2}(F)$ vertices, rather than part of them. For this sanity check to work, we prove that the equality holds if and only if $H = Deg_{\geq 2}(F)$.

To begin with, we prove that the equality holds if $H = Deg_{\geq 2}(F)$. This is trivial as for every graph G, we have the following relationship between its edge count, m, and its total degree count.

$$2m = \sum_{v\in V(G)} d(v) = n - |Deg_{\geq 2}(G)| + \sum_{v\in Deg_{\geq 2}(G)} d(v)$$

Next, we prove that the equality does not hold if $H \neq Deg_{\geq 2}(F)$. Suppose $H \neq Deg_{\geq 2}(F)$, note that by our argument above, it is impossible to have x such that $x \in H$ but $x \notin Deg_{\geq 2}(F)$. Hence, we only consider the case where $\exists x \in Deg_{\geq 2}(F)$ s.t. $x \notin H$. We have

$$\begin{aligned}
&2m - (n - |H| + \sum_{v\in H} d(v)) \\
&= n - |Deg_{\geq 2}(F)| + \sum_{v\in Deg_{\geq 2}(F)} d(v) - (n - |H| + \sum_{v\in H} d(v)) \\
&= \sum_{v\in Deg_{\geq 2}(F)} d(v) - |Deg_{\geq 2}(F)| - \sum_{v\in H} d(v) + |H| \\
&= \sum_{v\in Deg_{\geq 2}(F)\setminus H} d(v) - |Deg_{\geq 2}(F)| - \sum_{v\in H\setminus Deg_{\geq 2}(F)} d(v) + |H| \\
&= \sum_{v\in Deg_{\geq 2}(F)\setminus H} d(v) - |Deg_{\geq 2}(F)| - |H \setminus Deg_{\geq 2}(F)| + |H| \\
&= \sum_{v\in Deg_{\geq 2}(F)\setminus H} d(v) - |Deg_{\geq 2}(F) \setminus H| \\
&> 0
\end{aligned}$$

where the second-last equality holds because by our definition of $Deg_{\geq 2}(F)$, all non-leaf vertices are in $Deg_{\geq 2}(F)$, hence vertices in $H \setminus Deg_{\geq 2}(F)$ are leaves and have degree of 1. And the last inequality holds because vertices in $Deg_{\geq 2}(F)$ have degree at least 2, and $Deg_{\geq 2}(F) \setminus H \neq \emptyset$. Hence, if some $Deg_{\geq 2}(F)$ vertices are not sampled, i.e., $Deg_{\geq 2}(F) \setminus H \neq \emptyset$, then $2m \neq n - |H| + \sum_{v \in H} d(v)$.

It remains to prove that conditional on all $Deg_{\geq 2}(F)$ vertices being sampled, both $|\widehat{S}|$ and $|\widehat{Deg_{\geq 2}}|$ are exact estimates of $|Supp(F)|$ and $|Deg_{\geq 2}(F)|$ respectively. If all $Deg_{\geq 2}(F)$ vertices are sampled successfully, we can infer whether a vertex is a leaf or not by testing whether it is in H or not. As indicated by line 16 – 19, by counting the number of leaves, l_v, for each $v \in H$, we can identify all the support vertices in H. But not all support vertices have degree greater than 1, by definition we can have support vertices of degree 1 in paths of length 2 (P_2), such that each P_2 has two support vertices. Thus, we need to also count the number of P_2, p. This is trivial as if an edge arrives with none of its endpoints in $Deg_{\geq 2}(F)$, this edge must be an instance of P_2.

Lastly, the space used by Algorithm 4 is $\widetilde{\mathcal{O}}(K_3)$. In the first pass, the sparse recovery structure takes $\widetilde{\mathcal{O}}(K_3)$ space to store. And in the second pass, for each vertex in $Supp(F)$, we introduce two counters that can be both stored in $\mathcal{O}(\log n)$ bits. Therefore the total space usage of Algorithm 4 is $\widetilde{\mathcal{O}}(K_3)$.

◀

C.11 Proof of Theorem 32

Proof of Theorem 32. By Theorem 18 and Corollary 21, the min between $\frac{3(n+|Deg_1(F)|)}{8}$ and $\frac{n+|Deg_1(F)|-|Supp(F)|}{2}$ is guaranteed to be a 4/3 approximation of the independence number $\beta(F)$. Hence, it suffices to show that we could either approximate both terms well, or approximate the smaller term well.

When $|Deg_{\geq 2}(F)| \leq K_2 = 8\sqrt{n}$, the algorithm returns at line 11. By Lemma 31, Subroutine 4 gives exact estimates of $|Deg_{\geq 2}(F)|$ and $|Supp(F)|$ with probability $1 - 1/c_2 = 1 - \delta$. Also, since $n = |Deg_{\geq 2}(F)| + |Deg_1(F)|$ and n is known, we can obtain an exact estimate of $|Deg_1(F)|$. Hence, no matter which estimate is returned, it is a 4/3 approximation of $\beta(F)$.

When $|Deg_{\geq 2}(F)| > 8\sqrt{n}$, there are two cases to consider. If $|Supp(F)| \geq \sqrt{n}$, we may return the estimate of $\frac{3(n+|Deg_1(F)|)}{8}$ or the estimate of $\frac{n+|Deg_1(F)|-|Supp(F)|}{2}$ (line 8). On the one hand, if the estimate of $\frac{3(n+|Deg_1(F)|)}{8}$ is returned, by Lemma 28 we know that the algorithm on line 4 outputs a $(1 \pm \frac{\varepsilon}{2})$ estimate of $|Deg_1(F)|$ with probability $1 - \delta/2$. Thus our result is a $4/3(1 \pm \varepsilon/2)$ approximation of β.

On the other hand, if the estimate of $\frac{n+|Deg_1(F)|-|Supp(F)|}{2}$ is returned, by Lemma 30, Subroutine 3 gives a $(1 \pm \frac{\varepsilon}{2})$ estimate of $|Supp(F)|$ with probability

$$1 - 3e^{-c_1/3} = 1 - 3e^{-\ln(6\delta^{-1})} = 1 - \frac{\delta}{2}.$$

Applying a union bound over successfully returning an estimate of $|Supp(F)|$ and an estimate of $|Deg_1(F)|$, the probability of failure is at most δ. And on the upper-bound side, the approximation ratio is at most

$$
\begin{aligned}
&\frac{1}{2}\left(n + (1 \pm \frac{\varepsilon}{2})|Deg_1(F)| - (1 \pm \frac{\varepsilon}{2})|Supp(F)|\right) \\
&\leq \frac{1}{2}\left(n + |Deg_1(F)| - |Supp(F)| + \frac{\varepsilon}{2}\left(|Deg_1| + |S|\right)\right) \\
&\leq (1+\varepsilon)\,\frac{1}{2}\left(n + |Deg_1(F)| - |Supp(F)|\right)
\end{aligned}
$$

where the last inequality holds because $|Supp(F)| \leq |Deg_1(F)| \leq n$, thus $|Deg_1(F)| + |Supp(F)|$ is no more than $2(n+|Deg_1(F)|-|Supp(F)|)$. Similarly, we prove the approximation ratio on the lower bound as

$$
\begin{aligned}
&\frac{1}{2}\left(n + (1 \pm \frac{\varepsilon}{2})|Deg_1(F)| - (1 \pm \frac{\varepsilon}{2})|Supp(F)|\right) \\
&\geq (1-\varepsilon)\,\frac{1}{2}\left(n + |Deg_1(F)| - |Supp(F)|\right)
\end{aligned}
$$

Hence, our algorithm returns a $4/3 \cdot (1 \pm \varepsilon)$ approximation of $\beta(F)$.

Lastly, if $|Supp(F)| < \sqrt{n}$, then by Lemma 30, $|\widehat{S}| \leq 2\sqrt{n}$ with probability $1 - 2e^{-\frac{c_1}{3\varepsilon^2}}$. By Theorem 24, we know that

$$
|\widehat{Deg_{\geq 2}}| \geq (1-\varepsilon) \cdot |Deg_{\geq 2}(F)| > \frac{|Deg_{\geq 2}(F)|}{2}
$$

holds with probability $1-\delta/2$ (if $\varepsilon < 1/2$). Since $|Deg_{\geq 2}(F)| \geq 8\sqrt{n}$, by applying a union bound, we can claim that the probability of $2|\widehat{S}| > |\widehat{Deg_{\geq 2}}|$ is at most $2e^{-\frac{c_1}{3\varepsilon^2}} + \delta/2 \leq \delta$. Therefore, with probability $1-\delta$, the minimum between the two estimates is $\frac{3(n+|\widehat{Deg_1(F)}|)}{8}$. As shown before, $|\widehat{Deg_1(F)}|$ is a $1 \pm \varepsilon$ estimate of $|Deg_1(F)|$, hence the return estimate is a $4/3 \cdot (1 \pm \varepsilon)$ approximation of $\beta(F)$ with probability $1-\delta$.

The space usage of Algorithm 5 is the maximum space usage of its three sub-algorithms, which is $\widetilde{\mathcal{O}}(\sqrt{n})$.

◀

C.12 Algorithm for Estimating Domination Number in Forests

Algorithm 6 Main Algorithm for Estimating γ

Input: error rate ε and fail rate δ
Initialization: Set $K_1 = \sqrt{n}$, $K_2 = 12\sqrt{n}$, $c_1 = 3\ln(6\delta^{-1})$, $c_2 = \delta^{-1}$, $\varepsilon_1 = \varepsilon$
Run the following algorithms concurrently:
1. Algorithm for estimating $|Deg_{\geq 2}|$ with ε and $\delta/2$, denote the returned result as $|\widehat{Deg_{\geq 2}}|$
2. Subroutine 3 with K_1, c_1, and ε_1, denote the returned result as $|\widehat{S}|$
3. Subroutine 4 with K_2 and c_2, denote the returned result as $|\widehat{S}|'$ and $|\widehat{Deg_{\geq 2}}|'$
if Subroutine 4 returns $FAIL$ **then**
    Return $\widehat{\gamma} = \max\{\frac{2|\widehat{Deg_{\geq 2}}|}{3}, \frac{|\widehat{Deg_{\geq 2}}|+|\widehat{S}|}{2}\}$
else
    Return $\widehat{\gamma} = \max\{\frac{2|\widehat{Deg_{\geq 2}}|'}{3}, \frac{|\widehat{Deg_{\geq 2}}|'+|\widehat{S}|'}{2}\}$

Proof of Theorem 33. By Theorem 19 and Corollary 22, the max between $\frac{2|Deg_{\geq 2}(F)|}{3}$ and $\frac{|Deg_{\geq 2}(F)|+|Supp(F)|}{2}$ is guaranteed to be a 2 approximation of the domination number $\gamma(F)$. Thus it suffices to show that we could approximate both terms well, or we could approximate the larger term well.

When $|Deg_{\geq 2}(F)| \leq K_2 = 12\sqrt{n}$, by Lemma 31, Subroutine 4 succeeds with probability $1 - 1/c_2 = 1 - \delta$. Thus, by the design of the algorithm, we return our estimate at line 10. By Lemma 31, conditional on successful returning, Subroutine 4 gives exact estimates of $|Deg_{\geq 2}(F)|$ and $|Supp(F)|$. Hence we have a 2-approximation of $\gamma(F)$.

When $|Deg_{\geq 2}(F)| > 12\sqrt{n}$, there are two cases to consider. If $|Supp(F)| \geq \sqrt{n}$, then the algorithm might return the estimate of $\frac{2|Deg_{\geq 2}(F)|}{3}$ or the estimate of $\frac{|Deg_{\geq 2}(F)|+|Supp(F)|}{2}$. On the one hand, if the estimate of $\frac{2|Deg_{\geq 2}(F)|}{3}$ is returned, by Theorem 24, we know that the algorithm on line 4 outputs a $(1 \pm \varepsilon)$ estimate of $|Deg_{\geq 2}(F)|$ with probability $1 - \delta/2$. Hence we obtain a $2(1 \pm \varepsilon)$ approximation of γ.

On the other hand, if the estimate of $\frac{|Deg_{\geq 2}(F)|+|Supp(S)|}{2}$ is returned by our algorithm, we know that by Lemma 30, Subroutine 3 gives a $(1 \pm \varepsilon)$ estimate of $|Supp(S)|$ with probability

$$1 - 3e^{-c_1/3} = 1 - 3e^{-\ln(6\delta^{-1})} = 1 - \frac{\delta}{2}$$

Apply a union bound over successfully returning an estimate of $|Supp(F)|$ and an estimate of $|Deg_{\geq 2}(F)|$, the probability of failure is at most δ. And the error rate is still $1 \pm \varepsilon$ as we are summing up the two terms. Hence, the returned estimate is still a $2(1 \pm \varepsilon)$ approximation of γ.

Lastly, if $|Supp(F)| < \sqrt{n}$, then by Lemma 30, $|\widehat{S}| \leq 2\sqrt{n}$ with probability $1 - 2e^{-\frac{c_1}{3}}$. By Theorem 24, if $\varepsilon < 1/2$, then

$$|\widehat{Deg_{\geq 2}}| \geq (1 - \varepsilon)|Deg_{\geq 2}(F)| > \frac{|Deg_{\geq 2}(F)|}{2}$$

with probability $1 - \delta/2$. Since we have $|Deg_{\geq 2}(F)| > 12\sqrt{n}$, we can bound the probability of $3|\widehat{S}| \leq |\widehat{Deg_{\geq 2}}|$ as

$$\begin{aligned}
\Pr\left[3|\widehat{S}| \leq |\widehat{Deg_{\geq 2}}|\right] &= \Pr\left[|\widehat{S}| \leq 2\sqrt{n} \wedge |\widehat{Deg_{\geq 2}}| > 6\sqrt{n}\right] \\
&\geq 1 - \Pr\left[|\widehat{S}| > 2\sqrt{n}\right] - \Pr\left[|\widehat{Deg_{\geq 2}}| \leq 6\sqrt{n}\right] \\
&\geq 1 - 2e^{-\frac{c_1}{3}} - \delta/2 \\
&\geq 1 - \delta
\end{aligned}$$

where we apply the union bound in the first inequality. Therefore, with probability at least $1 - \delta$, the maximum between the two estimates, $\frac{2|\widehat{Deg_{\geq 2}}|}{3}$ and $\frac{|Deg_{\geq 2}|+|S|}{2}$, is $\frac{2|\widehat{Deg_{\geq 2}}|}{3}$. As shown before, $|\widehat{Deg_{\geq 2}}|$ is a $1 \pm \varepsilon$ estimate of $|Deg_{\geq 2}(F)|$, hence the returned estimate is a $2 \cdot (1 \pm \varepsilon)$ approximation of $\gamma(F)$ with probability $1 - \delta$.

The space usage of Algorithm 6 is the maximum space usage of its three sub-algorithms, which is $\widetilde{\mathcal{O}}(\sqrt{n})$.

◀