Simulating Random Walks in Random Streams

John Kallaugher* Michael Kapralov† Eric Price‡

Abstract

The random order graph streaming model has received significant attention recently, with problems such as matching size estimation, component counting, and the evaluation of bounded degree constant query testable properties shown to admit surprisingly space efficient algorithms.

The main result of this paper is a space efficient single pass random order streaming algorithm for simulating nearly independent random walks that start at uniformly random vertices. We show that the distribution of k-step walks from b vertices chosen uniformly at random can be approximated up to error ε per walk using $(1/\varepsilon)^{O(k)}2^{O(k^2)} \cdot b$ words of space with a single pass over a randomly ordered stream of edges, solving an open problem of Peng and Sohler [SODA '18]. Applications of our result include the estimation of the average return probability of the k-step walk (the trace of the k^{th} power of the random walk matrix) as well as the estimation of PageRank. We complement our algorithm with a strong impossibility result for directed graphs.

*UT Austin
†EPFL
‡UT Austin

Contents

1 Introduction

The random order streaming model for computation on graphs has been the focus of much attention recently, resulting in truly sublinear algorithms for several fundamental graph problems, i.e. algorithms whose space complexity is sublinear in the number of *vertices* (as opposed to edges) in the input graph [KKS14, CJMM17, MMPS17, PS18, KMNT20]. This is in sharp contrast to adversarially ordered streams, where $\Omega(n)$ space is often needed to solve even the most basic computational problems on graphs [FKM$^+$04]. This brings several fundamental problems on graph streams (matching size, number of connected components, constant query testable properties in bounded degree graphs) into the same regime as basic statistical queries such as heavy hitters, frequency moment estimation and distinct elements [AMS96]—problems that can be solved using space polylogarithmic in the length of the stream.

Sampling random walks has numerous applications in large graph analysis (e.g., [ST13, ACL06, AP09, COP03]), so it has received quite a bit of attention in the adversarial streaming model [SGP11, Jin19, CKP$^+$21]. However, while these results are useful for dense graphs, they all require $\Omega(n)$ space.

We show that the random order model allows us to break this barrier. For random order streams we give an algorithm that generates b walks that are ε-approximate to k-step random walks from uniformly random starting vertices[1], using $(\frac{1}{\varepsilon})^{O(k)} \cdot 2^{O(k^2)} \cdot b$ words of space, independent of the graph size n. This solves an open problem of Peng and Sohler on estimating return probabilities of random walks ([PS18], page 23).

The exponential dependence on poly(k) here seems likely to be necessary, at least up to the power of k, as recent work [CKKP21] has shown that finding a length-ℓ component in a graph where every component is of length at most ℓ requires $\ell^{\Omega(\ell)}$ space in a model close[2] to random-order streaming. Performing a $k = \Theta(\ell^2)$ random walk from a randomly chosen vertex would suffice for this, and so we expect any such algorithm needs at least $k^{\Omega(\sqrt{k})}$ space.

Our algorithm immediately implies sublinear algorithms for graph analytics based on short random walks, such as return probability estimation or PageRank. Consider PageRank with a constant reset probability α. For a "topic" $T \subset V$—think, "news websites" or "websites about gardening"—we can view the total PageRank of T as a measure of the importance of that topic to the graph. Our walk sampling algorithm lets us estimate the total PageRank of T to within ε using $O_{\alpha,\varepsilon}(1)$ space.

Our algorithmic results are for undirected graphs, because directed graphs are hard: we show that sampling walks from directed graphs (or just estimating PageRank) requires $\Omega(n)$ space in random order streams.

Our results. We now state our results formally. We will need the definition of ε-closeness of distributions below:

Definition 1.1. (Pointwise ε-closeness of distributions) *We say that a distribution $p \in \mathbb{R}_+^{\mathcal{U}}$ is ε-close pointwise to a distribution $q \in \mathbb{R}_+^{\mathcal{U}}$ if for every $u \in \mathcal{U}$ one has*

$$p(u) \in [1-\varepsilon, 1+\varepsilon] \cdot q(u).$$

We now define the notion of an ε-approximate sample of a k-step random walk:

Definition 1.2. (ε-approximate sample) *Given $G = (V, E)$ and a vertex $u \in V$ we say that $(X_0, X_1, \ldots, X_k)$ is an* ε-approximate sample *of the k-step random walk started at u if the distribution of $(X_0, X_1, \ldots, X_k)$ is ε-close pointwise to the distribution of the k-step walk started at u (see Definition 1.1).*

Main result. Our main result is an algorithm for generating nearly independent ε-approximate samples of the k-step walk in the input graph G presented as a randomly ordered stream:

[1] Previous work has considered this problem when the start vertex is adversarially chosen and given to the algorithm before processing the stream. Unfortunately $o(n)$ space is impossible in this setting, as if the start vertex is in, say, a two-edge path, finding the second edge of the path will be difficult in the 50% of cases it arrives after the first. For a formal lower bounds see Appendix E.

[2] This lower bound applies when edges are grouped into pairs, and the pairs arrive in a uniformly random order. It does not necessarily imply lower bounds on fully random-order streaming, but it seems unlikely that this particular structure would make the problem dramatically harder.

THEOREM 1.3. *There exists a constant $c' > 0$ such that for every n-vertex graph G, for every $\varepsilon \in (n^{-1/1000}, 1/2)$, $b \le n^{1/100}$, and $1 \le k \le \min\left\{\frac{c' \log n}{\log(1/\varepsilon)}, \sqrt{c' \log n}\right\}$, the following holds:*

The output of SIMULATEWALKS(k, ε, b) *(Algorithm 4 below) is $(n^{-1/100} + 2^{-b})$-close in TV distance to the distribution of b independent ε-approximate samples of the k-step random walk in G started at vertices chosen uniformly at random. The space complexity of* SIMULATEWALKS(k, ε, b) *is upper bounded by $(1/\varepsilon)^{O(k)} \cdot 2^{O(k^2)} \cdot b$.*

Using random walk sampling as a primitive, we give algorithms for two important graph problems: computing the average return probability of k-step random walks and estimating the PageRank of a subset of nodes.

Return probability estimation. For every integer $k \ge 1$ and $u \in V$ let $p_u^k \in \mathbb{R}^V$ denote the distribution of the simple k-step random walk started at u. The average return probability of k-step random walks in G is

$$\mathrm{rp}(G) = \frac{1}{n} \sum_{u \in V} p_u^k(u). \tag{1.1}$$

We say that $\widehat{\mathrm{rp}}(G)$ estimates $\mathrm{rp}(G)$ with precision $\varepsilon \in (0, 1)$ if

$$|\mathrm{rp}(G) - \widehat{\mathrm{rp}}(G)| \le \varepsilon. \tag{1.2}$$

REMARK 1.4. *Note that if the input graph G consists of disjoint connected components with mixing time bounded by $o(k)$, then $rp(G)$ is very close to the number of connected components in G. In particular, if every component in G has size at most q, the mixing times are bounded by $q^{O(1)}$, so this gives another algorithm for approximately counting the number of connected components in G using space $2^{poly(q)}$, which is comparable to [PS18].*

In general, the average return probability can be viewed as a more robust measure of connectivity than the number of components.

Our algorithm for approximating the average return probability is APPROXRP (Algorithm 1 below).

Algorithm 1 APPROXRP: approximate average k-step return probability over $u \in V$

procedure APPROXRP(k, ε) ▷ k is the desired walk length, $\varepsilon \in (0, 1)$ is a precision parameter
  $b \leftarrow \frac{D}{\varepsilon^2}$ for a sufficiently large constant $D > 0$
  $(v^i)_{i \in [b]} \leftarrow$ SIMULATEWALKS(k, ε, b)
  return $\frac{1}{b} \cdot |\{i :\in [b] : v_k^i = v_0^i\}|$ ▷ Empirical return probability
end procedure

COROLLARY 1.5. *There exists a constant $c' > 0$ such that for every graph $G = (V, E), |V| = n, |E| = m$, for every $\varepsilon \in (n^{-1/1000}, 1/2)$ and $1 \le k \le \min\left\{\frac{c' \log n}{\log(1/\varepsilon)}, \sqrt{c' \log n}\right\}$ the following conditions hold.*

Algorithm APPROXRP(ε, k) *(Algorithm 1 below) computes an ε-approximation to the average return probability of a k-step random walk in G given as a random order stream using space $(1/\varepsilon)^{O(k)} \cdot 2^{O(k^2)}$ with probability at least $9/10$ over the randomness of the stream and its internal randomness.*

The proof of Corollary 1.5 follows from Theorem 1.3 by standard concentration inequalities and is presented in Appendix C.

Estimating PageRank. For a reset probability $\alpha \in (0, 1)$ the PageRank vector with reset probability α, denoted by $p_\alpha \in \mathbb{R}^V$, satisfies

$$p_\alpha = \alpha \cdot \frac{\mathbb{1}}{n} + (1 - \alpha) M p_\alpha,$$

where M is the random walk transition matrix of G. We give an algorithm (APPROXPAGERANK, Algorithm 2 below) that, given a membership oracle for a subset $T \subseteq V$, computes an approximation $\widehat{p}_\alpha(T)$ to $p_\alpha(T) = \sum_{u \in T} p_\alpha(u)$ such that

$$|\widehat{p}_\alpha(T) - p_\alpha(T)| \le \varepsilon. \tag{1.3}$$

Our algorithm exploits the fact that PageRank with reset probability α is a mixture of distributions of random walks started with the uniform distribution over vertices of G whose length follows the geometric distribution with parameter α. Specifically,

$$p_\alpha = (I - (1-\alpha)M)^{-1} \cdot \alpha \frac{\mathbb{1}}{n} = \sum_{k \geq 0} \alpha(1-\alpha)^k M^k \cdot \frac{\mathbb{1}}{n}$$

Therefore, an additive ε-approximation to the PageRank $p_\alpha(T)$ of a set T as per (1.3) can be obtained by truncating the sum above to its first $O(\frac{1}{\alpha}\log(1/\varepsilon))$ terms and estimating them using SimulateWalks, our algorithm for generating independent samples of random walks. This is exactly what ApproxPageRank (Algorithm 2 below) does.

Algorithm 2 ApproxPageRank: approximate PageRank with reset probability $\alpha \in (0,1)$ on target set T up to ε additive error.

procedure ApproxPageRank(α, T, ε) ▷ Approximate PageRank of $T \subseteq V$
  ▷ α is the reset probability, ε is the additive precision
  ▷ T is given by a membership oracle
  $b \leftarrow \frac{D}{\varepsilon^2}$ for a sufficiently large constant $D > 0$
  $(v^i)_{i \in [b]} \leftarrow$ SimulateWalks($\lceil \frac{2}{\alpha}\log(1/\varepsilon)\rceil, \varepsilon, b \cdot \lceil \frac{2}{\alpha} \cdot \log(1/\varepsilon)\rceil$)
  ▷ Increase the number of sampled walks to account for different lengths
  for $j = 0$ to $\lceil \frac{2}{\alpha}\log(1/\varepsilon)\rceil$ **do**
    $W(j) \leftarrow$ walks v^i with i between $b \cdot j + 1$ and $b \cdot j$ truncated to j steps.
    ▷ $W(j)$ are nearly independent collections of s walks of length j
  end for
  $\widehat{p} \leftarrow 0$
  for $i = 1$ to b **do**
    $J \leftarrow \mathrm{Geom}(\alpha)$ ▷ $J = j$ with probability $\alpha(1-\alpha)^j$ for every integer $j \geq 0$
    If $J > \lceil \frac{2}{\alpha}\log(1/\varepsilon)\rceil$ **then continue**
    if i^{th} walk in $W(J)$ ends in T **then** ▷ Use membership oracle for T
      $\widehat{p} \leftarrow \widehat{p} + \frac{1}{b}$
    end if
  end for
  return $\widehat{p}$
end procedure

Corollary 1.6. *There exists a constant $c' > 0$ such that for every graph $G = (V, E), |V| = n, |E| = m$, for every $\alpha \in (0,1)$, every $\varepsilon \in \left(2^{-o(\sqrt{\log n})}, 1/2\right)$ such that $\frac{1}{\alpha} \leq \frac{\sqrt{\log n}}{4\log(1/\varepsilon)}$ the following conditions hold.*

For every $T \subseteq V$ ApproxPageRank (Algorithm 2) approximates $p_\alpha(T)$ for constant $\alpha \in (0,1)$ up to additive error ε with probability at least $9/10$ using $(1/\varepsilon)^{O(\frac{1}{\alpha}\log(1/\varepsilon))} \cdot 2^{O(\frac{1}{\alpha^2}\log^2(1/\varepsilon))}$ space given a randomly ordered stream of edges of G, assuming a membership oracle for the target set T.

The proof of Corollary 1.6 follows from Theorem 1.3 by standard concentration inequalities and is presented in Appendix C.

Lower bounds for directed graphs. PageRank was first studied for *directed* graphs, and so it is natural to ask if it is possible to extend these algorithms to that setting. We show that it is not, and in fact both sampling from the random walk distribution and approximating the PageRank of a vertex set in a directed graph require $\Omega(n)$ bit of storage. This holds even if we restrict to approximating the distribution of very short random walks.

Theorem 1.7. *For any constant $\varepsilon < 1/4$, the following holds for all $k \geq 3$ and all n: there is a family of directed graphs with no more than n vertices and edges such that any random order streaming algorithm that ε-approximates the distribution of length-k random walks on graphs drawn from the family uses $\Omega(n)$ bits of space, with a constant factor depending only on ε.*

THEOREM 1.8. *Let α be a given (constant) reset probability for PageRank. For any constants $\varepsilon < (1-\alpha)^3 - \frac{1}{2}$, $\delta < 1/4$, the following holds for all n: there is a family of directed graphs with no more than n vertices and edges such that any random order streaming algorithm that returns a ε additive approximation to the PageRank of vertex sets in these graphs with probability $1-\delta$ uses $\Omega(n)$ bits of space, with a constant factor depending only on α, ε, and δ.*

Note that this second lower bound applies only when $(1-\alpha)^3 > 1/2$, i.e. α must not be much greater than 0.2. In applications α is typically 0.15, so this is a reasonable assumption.

1.1 Algorithmic Techniques In what follows we discuss the main challenges involved in obtaining Theorem 1.3 and the key ideas behind our approach. A natural approach would be to sample a large collection of vertices in the graph uniformly at random, and to then try simulating k-step walks from them using the random order of the stream. Most of these simulations will fail, but our hope is that with a reasonably large (specifically, $\varepsilon^{O(k)}2^{-O(k^2)}$) probability τ, we will successfully output a nearly-uniform random walk from each start vertex, and that, when we don't succeed, we will know that we failed.

How could we simulate a walk from a given vertex v using the random stream? One idea might be: starting with v, repeatedly take the next edge incident on the current vertex, and stop after k steps (or output $\perp$ if the stream terminates before the k^{th} step). The intuition is that a random walk has a $1/k!$ chance of appearing in the stream in order, but this approach has two problems. First, it never traverses the same edge twice, while a random walk has a good chance of doing so whenever it encounters a low-degree vertex. Second, even for walks where every edge is distinct, the probability that it outputs a given path is not proportional to the probability of that path arising as a random walk. For example, when starting at the endpoint of a path, a length-2 walk occurs with probability 1/2 and is sampled with probability 1/2 (if the adjacent edge precedes the next edge out); but when starting in the middle of a path, the *first* adjacent edge is more likely to precede the next edge out, increasing the probability of sampling a length-2 walk to 2/3.

Repeated edges. We fix the first issue by associating each walk $(e_1, e_2, \ldots, e_k)$ with a template π, a sequence of numbers that encodes the amount of 'backtracking' that the walk $(e_1, e_2, \ldots, e_k)$ does. Formally, we define:

DEFINITION 1.9. (WALK TEMPLATE) *A k-walk template is a tuple $\pi \in \Pi_k$, where $\Pi_k := [1] \times [2] \times [3] \times \ldots \times [k]$.*

DEFINITION 1.10. (A WALK CONFORMING WITH A TEMPLATE) *We say that a walk $e = (e_1, e_2, \ldots, e_k) \in E^k$* conforms with a template $\pi \in \Pi_k$ *if for every $j \in [k]$ one has*

$$\pi_j = \min\{i \in [k] : e_i = e_j\}.$$

Similarly, for $\ell \in [k]$ we say that a walk $e = (e_1, e_2, \ldots, e_\ell) \in E^\ell$ conforms with a template $\pi \in \Pi_k$ *if for every $j \in [\ell]$ one has*

$$\pi_j = \min\{i \in [\ell] : e_i = e_j\}.$$

Note that a walk $(e_1, e_2, \ldots, e_k)$ conforms with a template $\pi \in \Pi_k$ if and only if for every $j \in [k]$, π_j is the first time the edge e_j appears in the walk.

Because every walk $(e_1, e_2, \ldots, e_k)$ conforms with exactly one template, our random walk generation procedure can proceed by first sampling a template uniformly at random, and then generating a walk that conforms with that template.

Debiasing the estimate. To address the second issue with the naive approach—that the probability we find a given walk is not proportional to the random walk probability—we modify the walk procedure slightly to not always follow the next edge out of each vertex. Instead, we choose our first edge out of the start vertex v uniformly at random from the edges incident to v in the first η fraction of the stream, for a small parameter η; our second edge is chosen uniformly from the edges incident to this vertex in the second η fraction of the stream, and so on. But this is subject to conforming to the template—in step j, if $\pi_j \neq j$, we ignore the j^{th} η fraction of the stream and instead reuse the π_j^{th} edge we've already taken.

With this approach, we can correct differences in the probability of finding each walk. The probability that we find a given walk is (i) $1/k!$, the probability that we sample the right template, times (ii) an η factor for each distinct edge in the walk, the probability that the stream is such that it is *possible* for our algorithm's random choices to find this walk, times (iii) the probability that our algorithm makes the correct choices to find the walk. This last probability depends on the stream, being the product over steps of the inverse number of edges incident to the previous vertex in the appropriate η fraction of the stream. The key is that this probability p is known after we see the stream; so if we knew the *true* random walk probability q for this walk, we could rejection sample with probability proportional to q/p to output walks under the correct distribution.

So how can we estimate the correct probability q of a given random walk with small expected error? The random walk probability q is $\prod_{j=1}^{k} \frac{1}{d_{j-1}}$, where d_{j-1} is the degree of the $(j-1)^{\text{th}}$ vertex in the walk (with d_0 being the starting degree). We can estimate each d_j by watching the stream after we finish the sampling procedure; this will contain a $(1-k\eta)$ fraction of the stream. One might expect this to introduce an error of about $(1-k\eta)^k \approx e^{-k^2\eta}$ in our estimate of q, but in fact the error can be much larger because a vertex may be visited as many as $\Theta(k)$ times in a walk.

For a constant-degree vertex, there is an ηk chance that we will miss at least one of its edges, in which case our estimate will be off by a constant factor, which could in turn lead to a $2^{\Theta(k)}$ relative error in q if the vertex appears $\Theta(k)$ times in the walk, for $\eta 2^{\Theta(k)}$ expected relative error. For this reason we need to set $\eta < 2^{-\Theta(k)}$, which leads to the final $2^{-O(k^2)}$ term in τ, and thus the $2^{O(k^2)}$ term in our space complexity.

Repetition and near-independence. The above argument leads to an $O(k)$-space algorithm that, with probability $2^{-O(k^2)}$, outputs a nearly uniform random walk. To make this useful, we need to repeat it at least $s = 2^{O(k^2)}$ times so that we may actually find walks. The challenge here is that these are not independent repetitions: the output of the algorithm depends on the random order of the stream, which is shared by each copy of the algorithm.

Fortunately, the repetitions are *nearly* independent. Knowing the path taken in a given attempt to sample a walk tells us something about the arrival times of the other edges incident to the vertices visited in that walk, but it is independent of edges not incident to vertices on the path. If the graph had no high degree vertices—say, the maximum degree were $n^{1/4}$—this would be sufficient: the probability that any given walk visits a degree-d vertex is at most $kd/n < n^{-.7}$, so if $s < n^{.01}$ we will probably never visit two adjacent vertices.

For high-degree vertices we need a different analysis: a high-degree vertex v will with high probability have many edges as possibilities in each stage, so knowing the behavior of s other walks only has a small effect on which edges are likely to be followed after visiting v. Formally, we introduce a hybrid algorithm where the behavior on high degree vertices is independent of the stream, show that the distribution of the output of the original algorithm is close to that of the hybrid algorithm, then use the above argument for low degree vertices.

1.2 Lower Bound Techniques Our lower bound is based on the following property of directed graphs: if a vertex has high *in*-degree but low *out*-degree, that vertex can cause the vast majority of random walks in a graph to be "channeled" into one path. If this path has multiple edges any algorithm that estimates the random walk distribution or the PageRank vector will need to observe all of them, which is inherently difficult as later edges in the path will not be recognized as significant unless they arrive after all the earlier edges. This is depicted in Figure 1.

Formally, we give a method for encoding an instance of the Indexing problem in a graph stream, similar to techniques used in [CCM16]. In the Indexing problem, Alice has an n bit string x and Bob an index I. Alice must send Bob a message that allows him to guess the value of x_I. It is known that Alice must send $\Omega(n)$ bits if she wants to succeed at this task.

Ordinarily it is difficult to encode communication problems as random order graph streams, as the fact that the edges may arrive in any order makes it difficult to assign parts of the graph to different players. We evade this difficulty by making use of the fact that the indexing problem is hard even if the players are guaranteed a uniform distribution on their inputs *and* only have to succeed with probability $1/2 + \varepsilon$ for some constant ε.

In our method, Bob encodes his index as a single edge from a high in-degree vertex, and Alice encodes her

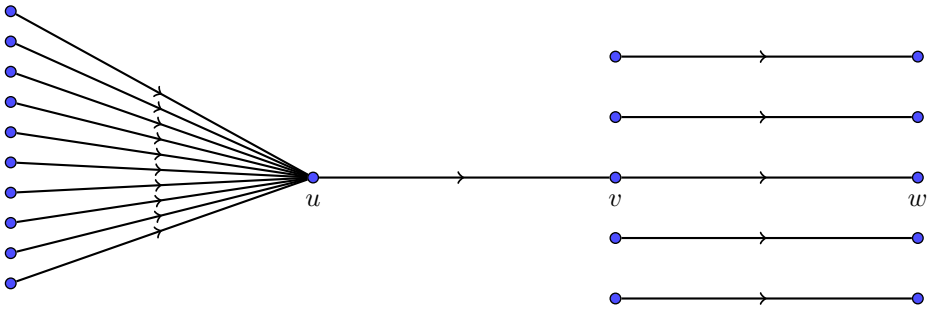


Figure 1: A directed graph that makes it hard to find walks. Most walks in the graph start at one of the vertices on the left and then go into u, then v, then w. But if vw arrives before uv it is impossible to know that it should be part of all of these walks.

bits as n edges that it might point to, with each pointing to a vertex representing 0 or 1. Therefore, almost all random walks in the graph end at a vertex representing x_I, and a large fraction of the PageRank vector's weight will be on this vertex.

As the edges are required to arrive in a uniformly random order, sometimes Alice's edges may arrive after Bob's edge, in which case Bob is responsible for inserting them in the stream. them. In that case he simply guesses what they should be. This means that half the time the graph will encode a random answer rather than a solution to Indexing, but this is still sufficient for him to succeed more than half the time. This encoding is illustrated in Figure 2.

1.3 Related Work For adversarial streams the problem of generating a k-step walk out of a given starting vertex was first considered in a paper of [SGP11], where it is shown how to generate n walks of length k using $\widetilde{O}(\sqrt{k})$ passes over the stream and $\widetilde{O}(n)$ space. The work of [Jin19] gives a single pass algorithm with space complexity $\widetilde{O}(n \cdot \sqrt{k})$ undirected graphs, and shows that this is best possible. Another recent work [CKP+21] gives two-pass algorithms for generating walks of length k in general (even directed) graphs using $\widetilde{O}(n \cdot \sqrt{k})$ space, which they also show it essentially best possible.

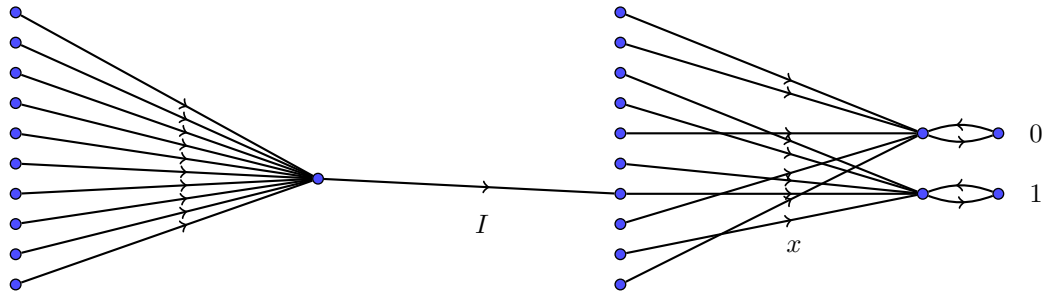
2 Basic Definitions and Claims

Basic notation. For any integer $a \geq 1$ we write $[a] = \{1, 2, \ldots, a\}$. For any set S we use $\mathcal{U}(S)$ to denote the uniform distribution on S. For a pair of (discrete) random variables X and Y we let $d_{TV}(X, Y)$ denote the total variation distance between X and Y, which equals one half of the ℓ_1 distance between their distributions. For a vertex $v \in V$ we write $d(v)$ to denote the degree of v in G, and $\delta(v)$ to denote the set of its incident edges. For an integer $k \geq 0$ and two vertices $u, v \in V$ we write $p_v^k(u)$ to denote the probability that the simple random walk started at v reaches u after k steps. We assume for simplicity that the graph does not have isolated vertices (all our algorithms can be easily adapted to handle isolated vertices, so this is without loss of generality).

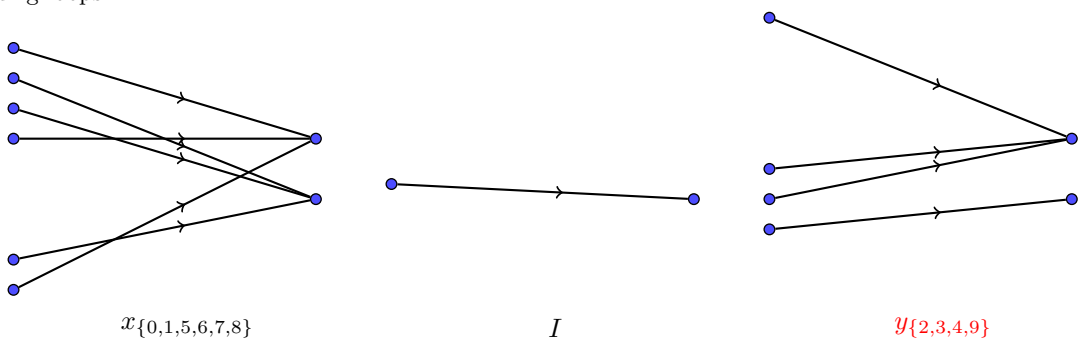
2.1 Stream Model We assume that we receive the edges of the graph in a uniformly random order. We think of this order as being generated by assigning each edge $f \in E$ a "timestamp" $t_f \in [0, 1]$ uniformly at random. The edges are then presented to the algorithm in ascending order of timestamp.

Typically a random-order stream does not come with such timestamps. However, as we show in Appendix B, an algorithm can generate timestamps distributed appropriately using just $O(\log n)$ extra space, for any desired $1/\operatorname{poly}(n)$ accuracy (this will suffice, as we can set the accuracy to e.g. $1/n^{100}$ and with very high probability the output of the algorithm will not be influenced by changing any number of timestamps by that much).

We assume knowledge of m, the number of edges in the graph. This assumption, however, can be removed by 'guessing' the right value of m (by running several copies of the algorithm in parallel), at the expense of a mild loss in the space complexity.



(a) Encoding an instance of Indexing in a directed graph. The high in-degree vertex will have one outgoing edge, encoding Bob's index I, which will point to one of n vertices, each of which encodes one of Alice's bits x by pointing to one of two corresponding loops.



(b) Converting the encoding into a random-order stream. The edges that do not depend on Alice or Bob's input can be inserted using shared randomness, and so are ignored here. Those of Alice's edges that should arrive *before* Bob's edge are encoded as normal, but those that arrive after are instead inserted by Bob by chosing a random guess y for Alice's input. Half of the time the edge encoding x_I arrives before the edge encoding I, and the other half of the time there is still a 0.5 chance of Bob guessing correctly, so the encoding is "correct" with probability 0.75.

Figure 2: Encoding an instance of Indexing as a random-order graph stream.

2.2 Algorithm Our walk sampling algorithm (SamplesWithReset, Algorithm 3 below) is quite simple: it samples a large enough set of nodes uniformly at random, together with independent uniformly random templates for every sampled node. It then runs our random walk generation procedure (WalkFromTemplate) from each such node, and outputs a sample of the runs that do not return FAIL (i.e., those invocations of WalkFromTemplate that terminate with a valid walk).

Algorithm 3 SamplesWithReset: simulate s samples of k-step walk started from uniformly random vertices, with reset (i.e., allowing walks to fail)

1: **procedure** SamplesWithReset(k, ε, s)
2: ▷ k is the desired walk length, $\varepsilon \in (0,1)$ is a precision parameter
3: $\eta \leftarrow \varepsilon^8 \cdot 2^{-Ck}$ ▷ For a large enough constant $C > 0$
4: $v_0^i \leftarrow$ independent uniform sample from V for $i \in [s]$
5: **for** $i \in [s]$ **do**
6: Choose $\pi^i \sim \mathcal{U}(\Pi_k)$ ▷ Π_k is the set of walk templates of length k, see Definition 1.9
7: $v^i \leftarrow$ WalkFromTemplate(v_0^i, π^i, k, η) ▷ Run in parallel on the same stream
8: **end for**
9: **return** $(v^i)_{i \in [s]}$
10: **end procedure**

Algorithm 4 SimulateWalks: simulate a samples of k-step walk started from uniformly random vertices

1: **procedure** SimulateWalks(ε, k, b)
2: ▷ k is the desired walk length, $\varepsilon \in (0,1)$ is a precision parameter
3: $s \leftarrow b \cdot 100\eta^{-k} \cdot k!$ ▷ Increase number of starting nodes to account for failed walks
4: $(v^i)_{i \in [s]} \leftarrow$ SamplesWithReset(k, ε, s)
5: **if** at least b of the walks $(v^i)_{i \in [s]}$ succeeded **then**
6: **return** the first b successful walks
7: **else**
8: **return** FAIL
9: **end if**
10: **end procedure**

Overview of random walk generation (WalkFromTemplate). Our main random walk generation procedure is WalkFromTemplate (Algorithm 5 below). The procedure gets as input a starting vertex, a template π, the target length of the walk and a parameter η that corresponds to the fraction of the stream that is used to generate a single step of the walk. Setting η small lets us limit various correlations, but hurts the performance, since the probability of WalkFromTemplate terminating with a valid walk (as opposed to outputting FAIL) is about η^k.

The procedure WalkFromTemplate itself is natural: it partitions the first $k \cdot \eta$ fraction of the stream into intervals of length η, and for every $j \in [k]$ either it uses the j^{th} interval to sample a new edge (if the template π prescribes this, i.e. satisfies $\pi_j = j$; see line 5 of Algorithm 5), or it takes the corresponding previously traversed edge if the template π prescribes this, i.e. satisfies $\pi_j < j$, and the corresponding edge is in the neighborhood of the current vertex (see line 11 of Algorithm 5). The sampling is done using reservoir sampling, and therefore is space efficient.

After k phases, having constructed a candidate walk $(f_1, f_2, \ldots, f_k)$, the algorithm first uses the empirical batches to infer a partition of the candidate walk into batches, and then uses this information to perform rejection sampling. The goal of rejection sampling is to reduce the probability of picking up a walk to be proportional to the product of the inverse degrees of the first k vertices in the walk, i.e. to the correct probability. To achieve this WalkFromTemplate maintains the probability p_k of having collected the walk $(f_1, \ldots, f_k)$. The algorithm then uses the remainder of the stream to compute estimates $(\widehat{d}_j)_{j=0}^{k-1}$, of the degrees of the vertices on the candidate

walk, and keeps the candidate walk with probability proportional to $p_k^{-1}\prod_{j=1}^{k-1}(1/\widehat{d}_j)$ (see line 20 of Algorithm 5).

The algorithm is formally described as Algorithm 5 below. For a stream σ and parameters $\alpha, \beta \in [0,1]$ we define

$$\sigma[\alpha,\beta) = \{e \in E : t_e \in [\alpha,\beta)\}.$$

Recall that we think of every edge $e \in E$ as being assigned an independent uniformly random timestamp $t_e \in [0,1]$, and the edges being presented in increasing order of these timestamps (see Section 2.1 for more discussion of this assumption).

Algorithm 5 WalkFromTemplate: generate a random walk from starting vertex u_0 that conforms with a template π

1: **procedure** WalkFromTemplate(u_0, π, k, η) ▷ k is the desired walk length, u_0 is the starting vertex
2: **for** $j = 1$ to k **do**
3: $u \leftarrow u_{j-1}$
4: **if** $\pi_j = j$ **then**
5: $H_j \leftarrow$ edges in $\delta(u) \cap \sigma[\eta\cdot(j-1), \eta\cdot j)$
6: **If** $H_j = \emptyset$ **then return** FAIL
7: $f_j \leftarrow \mathcal{U}(H_j)$ ▷ Implemented using reservoir sampling
8: $\gamma_j \leftarrow \frac{1}{|H_j|}\cdot\eta$
9: **else**
10: **If** $f_{\pi_j} \notin \delta(u)$ **then return** FAIL
11: $f_j \leftarrow f_{\pi_j}$
12: $\gamma_j \leftarrow 1$
13: **end if**
14: $u_j \leftarrow$ endpoint of f_j other than u
15: **end for**
16: **for** $j \in [k]$ **do** ▷ Compute degree estimates for vertices on the walk
17: $\widehat{d}_{j-1} \leftarrow$ degree of u_{j-1} in $\{f_1,\ldots,f_k\} \cup \sigma[\eta\cdot k, 1]$
18: **end for**
19: $\alpha \leftarrow \prod_{j\in[k]} \min(\frac{\eta}{\gamma_j \widehat{d}_{j-1}}, 1)$ ▷ Done in postprocessing
20: **return** $(u_0,\ldots,u_k)$ with probability α and **return** FAIL otherwise
21: **end procedure**

In what follows we prove that, if we set our parameters appropriately, each walk is output with almost the correct probability.

We assume that parameters n, k and η satisfy the following:

(P1) $k \le c\log n/\log\log n$ for a small constant $c > 0$

(P2) $\eta \in (n^{-1/100}, 2^{-Ck})$ for a sufficiently large constant $C \ge 8$

3 Analysis of WalkFromTemplate

Lemma 3.1. *For every integer $k \ge 1$ and real number η satisfying **(P1)** and **(P2)**, and every $\pi \in \Pi_k$, the following holds:*

For every $v \in V$ and every length k walk $\mathbf{v} = (v_0, v_1, \ldots, v_{k-1}, v_k)$ from $v_0 = v$, an invocation of WalkFromTemplate(v, π, k, η) *(Algorithm 5) outputs $\mathbf{v}$ with probability*

$$p \in \left[1, 1+\eta^{1/7}\right]\cdot\eta^k\cdot\prod_{j\in[k]}\frac{1}{d(v_{j-1})},$$

if $\mathbf{v}$ conforms with π and with probability 0 otherwise.

The following corollary is an immediate consequence of Lemma 3.1:

COROLLARY 3.2. *For every integer $k \geq 1$ and real number η satisfying **(P1)** and **(P2)**, the following holds for π sampled from $\mathcal{U}(\Pi_k)$:*

For every $v \in V$ and every length k walk $\mathbf{v} = (v_0, v_1, \ldots, v_{k-1}, v_k)$ from $v_0 = v$, an invocation of WALKFROMTEMPLATE(v, π, k, η) *(Algorithm 5) outputs $\mathbf{v}$ with probability*

$$p \in \left[1, 1 + \mathrm{O}\left(\eta^{1/7}\right)\right] \cdot \frac{1}{k!} \cdot \eta^k \cdot \prod_{j \in [k]} \frac{1}{d(v_{j-1})}.$$

Proof. Per Definition 1.10 there exists a unique template $\pi' \in \Pi_\ell$ that $\mathbf{v}$ conforms with. The walk $\mathbf{v}$ conforms with any extension π of π' to a template in Π_k, and no other template. For every such π the corresponding invocation of WALKFROMTEMPLATE(v_0, π, k, η) constructs $\mathbf{v}$ at the end of the first ℓ iterations of its main loop with probability

$$p \in \left[1, 1 + \mathrm{O}\left(\eta^{1/7}\right)\right] \cdot \eta^\ell \cdot \prod_{j \in [\ell]} \frac{1}{d(v_{j-1})} \tag{3.4}$$

by Lemma 3.1, and with probability zero for other π. Thus, the result follows since π is selected uniformly at random from Π_k. ∎

3.1 Useful Technical Results The following are results that will be useful in our analysis. Proofs are deferred to Appendix A.

Recall that for integer $k \geq 0$ and two vertices $u, v \in V$ we write $p_v^k(u)$ to denote the probability that the simple random walk started at v reaches u after k steps.

CLAIM 3.3. *Let $v \in V$ be chosen uniformly at random in a graph with no isolated vertices. Then for every $u \in V$ and $k \geq 0$ one has $\mathbb{E}_{v \sim \mathcal{U}(V)}[p_v^k(u)] \leq d(u)/n$.*

The following is a consquence of Bennett's inequality.

LEMMA 3.4. *Let $Y = \sum_i \alpha_i X_i$, where $\alpha_i \in \{0, 1\}$ and $X_i \sim Ber(\eta)$ are independent for some $\eta \in (0, 1/50)$. Then for every $d \geq \sum_i \alpha_i$*

$$\Pr[Y \geq d/2] \leq (3\eta)^{d/5}.$$

LEMMA 3.5. *Let $E_1, \ldots, E_k, Z_1, \ldots, Z_k$ be arbitrarily correlated random variables. Let $\widetilde{\eta} \in (0, e^{-5k})$ and the positive integers $(q_i)_{i=1}^k$ be such that, for all $i \in [k]$, $E_i \in \{0, 1\}$, $\mathbb{E}[E_i] \leq \widetilde{\eta}^{q_i/5}$, $Z_i \in [0, 1]$, and $\mathbb{E}[Z_i] \leq \widetilde{\eta}$. Then*

$$\mathbb{E}\left[\prod_{i=1}^k (1 + Z_i + q_i E_i)\right] \leq 1 + 3^k \widetilde{\eta}^{1/5}.$$

3.2 Proof of Lemma 3.1 We now prove Lemma 3.1, restated here for convenience of the reader.

LEMMA 3.6. *For every integer $k \geq 1$ and real number η satisfying **(P1)** and **(P2)**, and every $\pi \in \Pi_k$, the following holds:*

For every $v \in V$ and every length k walk $\mathbf{v} = (v_0, v_1, \ldots, v_{k-1}, v_k)$ from $v_0 = v$, an invocation of WALKFROMTEMPLATE(v, π, k, η) *(Algorithm 5) outputs $\mathbf{v}$ with probability*

$$p \in \left[1, 1 + \eta^{1/7}\right] \cdot \eta^k \cdot \prod_{j \in [k]} \frac{1}{d(v_{j-1})},$$

if $\mathbf{v}$ conforms with π and with probability 0 otherwise.

Proof. There are three sources of randomness in WalkFromTemplate (Algorithm 5): the stream σ, the reservoir sampling r to find the path, and the rejection sampling at the end. We first analyze the event $\mathcal{F}$ that a given walk $\mathbf{v}$ is "collected", meaning that is found but might be rejected in the final rejection sampling step. We use F to denote the indicator variable associated with $\mathcal{F}$.

For $j \in [k]$ let $e_j = (v_{j-1}, v_j)$. Let

$$\Gamma := \left\{\sigma \text{ a stream } : \forall j \in [k] \text{ such that } \pi_j = j \text{ one has } t_{e_j} \in [\eta(j-1), \eta \cdot j)\right\}. \tag{3.5}$$

This is the set of streams σ such that collecting $\mathbf{v}$ is possible: for every $\sigma \notin \Gamma$ we have $F = 0$ always. Similarly, if $\mathbf{v}$ does not conform to π then $F = 0$. For any fixed $\sigma \in \Gamma$ and $\mathbf{v}$ that conforms to π, we have

$$\Pr_r[\mathcal{F}] = \prod_{j \in [k]: \pi_j = j} \frac{1}{|H_j|}, \tag{3.6}$$

where (as in Algorithm 5) H_j is the set of edges that could could be taken in stage j.

Define

$$p := \prod_{j=1}^{k} \gamma_j = \eta^{|\pi|} \cdot \prod_{j \in [k]: \pi_j = j} \frac{1}{|H_j|} = \eta^{|\pi|} \Pr_r[\mathcal{F}], \tag{3.7}$$

where we let $|\pi| := |\{j \in [k] : \pi_j = j\}|$. We note that $|\pi| \le k$ for every π. This implies that, for any stream $\sigma \in \Gamma$ and π that $\mathbf{v}$ conforms to,

$$\mathop{\mathbb{E}}_r\left[\frac{1}{p}F\right] = \eta^{-|\pi|}.$$

Since $\Pr[\sigma \in \Gamma] = \eta^{|\pi|}$, this means

$$\mathop{\mathbb{E}}_{\sigma,r}\left[\frac{1}{p}F\right] = 1. \tag{3.8}$$

Define $\widehat{d}(v_j)$ to be the degree of v_j in $\mathbf{v} \cup \sigma[\eta k, 1]$, so $\widehat{d}_j = \widehat{d}(v_j)$ when $\mathcal{F}$ occurs, and define $q := \eta^k \prod_{i=1}^{k} \frac{1}{\widehat{d}(v_{j-1})}$ and $q^* := \eta^k \prod_{i=1}^{k} \frac{1}{d(v_{j-1})}$. Note that, for any $\mathbf{v}$, q^* depends only on the graph and is independent of the stream order and the randomness of the algorithm. If $\mathbf{v}$ is collected, then it is output with probability $\alpha = \prod_{j \in [k]} \min\left(\frac{\eta}{\gamma_j \widehat{d}_{j-1}}, 1\right)$, which satisfies

$$\frac{q^*}{p} \le \alpha \le \frac{q}{p}. \tag{3.9}$$

The upper bound follows from ignoring the $\min(\cdot, 1)$ in the expression, and the lower bound follows from the fact that $\gamma_j = \frac{\eta}{|H_j|} \ge \frac{\eta}{d(v_{j-1})}$ and $\widehat{d}(v_j) \le d(v_j)$ for all j, so

$$\min\left(\frac{\eta}{\gamma_j \widehat{d}(v_{j-1})}, 1\right) \ge \min\left(\frac{\eta}{\gamma_j d(v_{j-1})}, 1\right) = \frac{\eta}{\gamma_j d(v_{j-1})}.$$

We would like to bound the probability $\mathbf{v}$ is output over the streams and internal randomness, which is

$$\mathop{\mathbb{E}}_{\sigma,r}[\alpha F]. \tag{3.10}$$

Lower bound. By (3.9) and (3.8),

$$\mathop{\mathbb{E}}_{\sigma,r}[\alpha F] = \mathop{\mathbb{E}}_{\sigma,r}\left[\alpha p \frac{1}{p} F\right] \ge \mathop{\mathbb{E}}_{\sigma,r}\left[q^* \frac{1}{p} F\right] = q^* \tag{3.11}$$

as q^* is independent of the stream order and the randomness of the algorithm.

Upper bound. We have that

$$\mathop{\mathbb{E}}_{\sigma,r}[\alpha F] \le \mathop{\mathbb{E}}_{\sigma,r}\left[\frac{q}{p}F\right] = \mathop{\mathbb{E}}_{\sigma}\left[q \cdot 1_{\sigma\in\Gamma} \cdot \mathop{\mathbb{E}}_{r}\left[\frac{1}{p}F\middle|\sigma\right]\right] = \mathop{\mathbb{E}}_{\sigma}\left[q \cdot 1_{\sigma\in\Gamma} \cdot \eta^{-|\pi|}\right] = \mathop{\mathbb{E}}_{\sigma}[q|\sigma\in\Gamma].$$

So it suffices to show that q is not much bigger than q^* on average over streams in Γ.

For any given $i \in \{0,1,\dots,k-1\}$, consider the distribution of the degree estimate $\widehat{d}(v_i)$ over $\sigma \in \Gamma$. If v_i has ℓ distinct incident edges among the walk $\mathbf{v}$, then those edges will always count in $\widehat{d}(v_i)$. Every other edge will count if and only if its timestamp is at least $k\eta$, which is an independent binary random variable with expectation $1-k\eta$. Let Y_i denote the number of edges that do *not* count, so $\widehat{d}(v_i) = d(v_i) - Y_i$ and $Y_i \sim B(d(v_i)-\ell, k\eta)$. We now analyze

$$\mathop{\mathbb{E}}_{\sigma}\left[\frac{q}{q^*}\middle|\sigma\in\Gamma\right] = \mathop{\mathbb{E}}_{\sigma}\left[\prod_{i=0}^{k-1}\frac{d(v_i)}{\widehat{d}(v_i)}\middle|\sigma\in\Gamma\right] = \mathop{\mathbb{E}}_{\sigma}\left[\prod_{i=0}^{k-1}\frac{d(v_i)}{d(v_i)-Y_i}\middle|\sigma\in\Gamma\right].$$

Define $\mathcal{I}_i$ to be the event that $Y_i \ge d(v_i)/2$. By the Bennett inequality corollary Lemma 3.4, if the constant in **(P2)** is chosen to be large enough,

$$\Pr[\mathcal{I}_i] \le (3k\eta)^{d(v_i)/5}.$$

On the other hand, when $\mathcal{I}_i$ does not occur and $Y_i < d(v_i)/2$ we have

$$\frac{d(v_i)}{d(v_i)-Y_i} = \left(\sum_{j=0}^{\infty}(Y_i/d(v_i))^j\right) \le 1 + 2Y_i/d(v_i).$$

Let $Z_i = 2Y_i/d(v_i)$ when $\overline{\mathcal{I}_i}$ holds and 0 otherwise. Since $\widehat{d}(v_i) \ge 1$, we always have $\frac{d(v_i)}{\widehat{d}(v_i)} \le d(v_i)$. Therefore

$$\frac{d(v_i)}{\widehat{d}(v_i)} \le 1 + Z_i + d(v_i)I_i.$$

where I_i is the indicator for $\mathcal{I}_i$. We can now apply Lemma 3.5 (if the constant in **(P2)** is chosen to be large enough) to $(I_0,\dots,I_{k-1}),(Z_0,\dots,Z_{k-1})$, and $\widetilde{\eta} = 3k\eta$, to say that

$$\mathop{\mathbb{E}}_{\sigma}\left[\frac{q}{q^*}\middle|\sigma\in\Gamma\right] = \mathop{\mathbb{E}}_{\sigma}\left[\prod_{i=0}^{k-1}\frac{d(v_i)}{\widehat{d}(v_i)}\middle|\sigma\in\Gamma\right] \le 1 + 3^k(3k\eta)^{1/5}.$$

Therefore the probability we output $\mathbf{v}$ satisfies

$$\mathop{\mathbb{E}}_{\sigma,r}[\alpha F] \le \mathop{\mathbb{E}}_{\sigma}[q|\sigma\in\Gamma] = q^* \mathop{\mathbb{E}}_{\sigma}\left[\frac{q}{q^*}\middle|\sigma\in\Gamma\right] \le (1+3^k(3k\eta)^{1/5})q^*.$$

For $\eta < 2^{-Ck}$ for sufficiently large C, this is at most $(1+\eta^{1/7})q^*$ as desired. ∎

4 Near-Independence of Constructed Walks

In this section we prove that the walks constructed by SamplesWithReset(k,ε,s) (Algorithm 3) are close to independent in total variation distance. We introduce a useful modified version of SamplesWithReset and WalkFromTemplate, as well as some notation, before stating the key formal lemmas.

Hybrid SamplesWithReset **and** WalkFromTemplate **algorithms.** As a first step we show that the output distribution of SamplesWithReset (which relies on WalkFromTemplate) is close in total variation distance to the output distribution of a modified version SamplesWithResetHybrid (which in turn relies on a modified WalkFromTemplateHybrid) – see Algorithm 6 and Algorithm 8 below.

The main reason behind the introduction of this algorithm is that it will be easier to prove near-independence of several walks generated on the same stream by SamplesWithResetHybrid than to perform the same analysis directly on SamplesWithReset. The proof of closeness of the output of SamplesWithReset

and SamplesWithResetHybrid in distribution proceeds by the 'hybrid argument', and to facilitate this argument the procedure WalkFromTemplateHybrid takes a parameter j^*, and changes its behavior relative to WalkFromTemplate in the first j^* iterations of the outer loop. We note that WalkFromTemplateHybrid is not an actual algorithm that can be run on a stream (it uses information that is not available as the stream comes in, such as exact vertex degrees and exact vertex neighborhoods), and is only a useful construct that facilitates analysis.

Algorithm 6 SamplesWithResetHybrid: simulate s samples of k-step walk started from uniformly random vertices, with reset (i.e., allowing walks to fail)

procedure SamplesWithResetHybrid(k, ε, s, j^*)
  ▷ k is the desired walk length, $\varepsilon \in (0,1)$ is a precision parameter
  $\eta \leftarrow \varepsilon^8 \cdot 2^{-Ck}$ ▷ For a large enough constant $C > 0$
  $v_0^i \leftarrow$ independent uniform sample from V for $i \in [s]$
  for $i \in [s]$ **do**
    Choose $\pi^i \sim \mathcal{U}(\Pi_k)$ ▷ Π_k is the set of walk templates of length k, see Definition 1.9
    $v^i \leftarrow$ WalkFromTemplateHybrid$(v_0^i, \pi^i, k, \eta, j^*)$ ▷ Run in parallel on the same stream
  end for
  return $(v^i)_{i \in [s]}$
end procedure

Algorithm 7 SimulateWalksHybrid: : simulate b samples of k-step walk started from uniformly random vertices

procedure SimulateWalksHybrid(k, ε, b, j^*)
  ▷ k is the desired walk length, $\varepsilon \in (0,1)$ is a precision parameter
  $\eta \leftarrow \varepsilon^8 \cdot 2^{-Ck}$ ▷ For a large enough constant $C > 0$
  $s \leftarrow b \cdot 100\eta^{-k} \cdot k!$ ▷ Increase number of starting nodes to account for failed walks
  $v_0^i \leftarrow$ independent uniform sample from V for $i \in [s]$
  $(v^i)_{i \in [s]} \leftarrow$ SamplesWithResetHybrid(k, ε, s, j^*)
  if at least b of the walks $(v^i)_{i \in [s]}$ succeeded **then**
    return the first b successful walks
  else
    return FAIL
  end if
end procedure

We first define

$$L = \{u \in V : d(u) \leq n^{1/4}\} \tag{4.12}$$

to be the set of 'low degree' vertices in the graph (the threshold of $n^{1/4}$ is somewhat arbitrary, but in general this threshold cannot be too high; in particular, it needs to be bounded away from n by at least an s^2 factor, where s is the number of samples used in SamplesWithReset). For the first j^* iterations of the main loop WalkFromTemplateHybrid (Algorithm 8)

- if the current vertex u is a high degree vertex (belongs to $V \setminus L$), samples its next edge uniformly at random from $\delta(u)$, without using the stream (see line 6)
- when estimating vertex degrees, uses (appropriately scaled) exact degrees for high degree vertices (see line 21), for all $j \leq j^*$.

The changes introduced in WalkFromTemplate (Algorithm 5) with respect to WalkFromTemplateHybrid are highlighted in red in Algorithm 8.

There is one other difference in the implementations of WalkFromTemplate and WalkFromTemplateHybrid, which does not affect the output but is convenient for the analysis: in WalkFromTemplateHybrid, the rejection sampling is done progressively. The final acceptance probability α_k for a walk equals the acceptance probability α in WalkFromTemplate, and the intermediate acceptance probabilities α_j let us show that the intermediate states are spread out comparably to a uniform random walk (proved in Lemma 4.6).

This means that the chance a random walk is still accepted after j steps is

$$\alpha_j := \prod_{g=1}^{j} \min\left(\frac{\eta}{\gamma_g \widehat{d}^{j}_{g-1}}, 1\right)$$

The following lemma shows that the output distribution of SamplesWithReset(k, ε, s) is close to SamplesWithResetHybrid(k, ε, s, k) in total variation distance:

Lemma 4.1. *Assuming* ***(P1)***, ***(P2)*** *and* $s \leq n^{1/30}$, *for every input graph* G, *we have that the output of* SamplesWithReset(k, ε, s) *is* $O(s \cdot n^{-1/10})$*-close in total variation distance to the output of* SamplesWithResetHybrid(k, ε, s, k).

The proof of Lemma 4.1 is presented in Section 4.2 below.

Algorithm 8 WalkFromTemplateHybrid: generate a random walk from starting vertex u_0 that conforms with a template π, modified on high degree vertices

procedure WalkFromTemplateHybrid(v_0, π, k, η, j^*) ▷ k is the desired walk length, u_0 is the starting vertex
  for $j = 1$ to k **do**
    $u \leftarrow v_{j-1}$
    if $\pi_j = j$ **then**
      If $u \in V \setminus L$ and $j \leq j^*$ **then**
        $H_j \leftarrow \delta(u)$
      else
        $H_j \leftarrow$ edges in $\delta(u) \cap \sigma[\eta \cdot (j-1), \eta \cdot j)$
      If $H_j = \emptyset$ **then return** FAIL
      $f_j \leftarrow \mathcal{U}(H_j)$ ▷ Implemented using reservoir sampling
      If $u \in V \setminus L$ and $j \leq j^*$ **then** $\gamma_j \leftarrow \frac{1}{d(u)}$ **else** $\gamma_j \leftarrow \frac{1}{|H_j|} \cdot \eta$
    else
      If $f_{\pi_j} \notin \delta(u)$ **then return** FAIL
      $f_j \leftarrow f_{\pi_j}$
      $\gamma_j \leftarrow 1$
    end if
    $v_j \leftarrow$ endpoint of f_j other than u
    for $g \in [j]$ **do** ▷ Compute degree estimates for vertices on the walk
      $\widehat{d}^{j}_{g-1} \leftarrow$ degree of v_{g-1} in $\{f_1, \ldots, f_j\} \cup \sigma[\eta \cdot k, 1]$
      If $v_{g-1} \in V \setminus L$ and $g \leq j^*$ **then**
        $\widehat{d}^{j}_{g-1} \leftarrow (1 - k\eta) d(v_{g-1})$
    end for
    $\alpha_j \leftarrow \prod_{g=1}^{j} \min\{\frac{\eta}{\gamma_g \cdot \widehat{d}^{j}_{g-1}}, 1\}$ ▷ Observe that $\alpha_j \leq \alpha_{j-1}$
    return FAIL with probability $1 - \alpha_j/\alpha_{j-1}$
      ▷ Note that the chance of returning FAIL by step j is $1 - \alpha_j$.
  end for
  return $(v_0, \ldots, v_k)$
end procedure

4.1 Notation A length-ℓ "partial" walk $\mathbf{v} = (v_0, \ldots, v_\ell) \in (V \cup \{\perp\})^{\ell+1}$ consists of at least 1 vertex of a random walk, followed (possibly) by a series of $\perp$. A collection of s partial walks is $\vec{\mathbf{v}} = (\mathbf{v}^i)_{i\in[s]}$.

DEFINITION 4.2. (SAMPLED AND COLLECTED VERTICES) *We say that a partial walk* $\mathbf{v}$ *is "collected" by an execution of* WALKFROMTEMPLATEHYBRID$(k, \varepsilon, \eta, \ell)$*, if for each* j*,* v_j *is the value set by the algorithm or* $\perp$ *if the algorithm returns FAIL before setting* v_j*.*

We say that a partial walk $\mathbf{v}$ *is "sampled" by an execution of* WALKFROMTEMPLATEHYBRID *in the same situation, except that* $v_j = \perp$ *if the algorithm rejects* v_j *by returning FAIL in round* j*.*

For an invocation of SAMPLESWITHRESETHYBRID$(k, \varepsilon, s, \ell)$*, we say that* $\vec{\mathbf{v}}$ *is collected or sampled if* $(\mathbf{v}^1, \ldots, \mathbf{v}^s)$ *are sampled or generated by the* s *executions of* WALKFROMTEMPLATEHYBRID*, respectively.*

In other words, if the walk fails at the end of round j, the walk that is "collected" still includes v_j, while the walk that is sampled does not (if the walk succeeds, or if the algorithm returns FAIL in line 9 or 13, they are identical).

DEFINITION 4.3. *For a length-*ℓ *partial walk* $\mathbf{v}$ *we define its neighborhood*

$$\Psi(\mathbf{v}) = \bigcup_{0\le j\le \ell, v_j \in L} \delta(v_j).$$

For a collection of walks $\vec{\mathbf{v}}$ *we define*

$$\Psi(\vec{\mathbf{v}}) = \bigcup_{\mathbf{v}} \Psi(\mathbf{v}).$$

DEFINITION 4.4. *For* $j \in 0, 1, \ldots, k$ *let* $\mathcal{F}_j$ *denote the following random variables:*

1. *The partial walks* $\vec{\mathbf{v}}_{\le j-1} := (v^i_{\le j-1})_{i\in[s]}$ *sampled by* SAMPLESWITHRESETHYBRID(k, ε, s, j).
2. *Timestamps of edges in* $\Psi(\vec{\mathbf{v}}_{\le j-1})$*, i.e.*

$$\{(f, t_f) : f \in \Psi(\vec{\mathbf{v}}_{\le j-1})\}.$$

3. *The internal randomness of* SAMPLESWITHRESETHYBRID(k, ε, s, j) *(Algorithm 6) used up to step* $j - 1$.

We let $\mathcal{F}_0 = \emptyset$.

DEFINITION 4.5. *For a collection of walks* $\vec{\mathbf{v}}$ *and* $j \in [k]$ *we define* $E^*_j := \Psi(\vec{\mathbf{v}}_{\le j-1}) \cup \{(v^i_{j-1}, v^{i'}_{j-1}) \mid i, i' \in [s]\}$ *to contain all edges out of low degree vertices in the first* j *steps of any walk, combined with all edges between vertices visited at position* $j - 1$ *in the different walks.*

4.2 Proof of Lemma 4.1

LEMMA 4.6. *Let* $v \in V$ *uniformly at random and* $\pi \sim \mathcal{U}(\Pi_k)$*, and consider the execution of* WALKFROMTEMPLATEHYBRID$(v, \pi, k, \eta, 0)$*. We have for any* $u \in V$ *and* $j \in [k]$ *that*

$$\Pr[v_j = u \mid v_j \neq \perp] \le 2d(u)/n.$$

Proof. Let $p^j_v(u)$ be the probability that a j-step random walk from v ends at u. By Claim 3.3, $\mathbb{E}_{v\sim U(V)}\left[p^j_v(u)\right] \le d(u)/n$.

First, consider $j = k$. The distribution of v_k is the same as for the last vertex u_k output by WALKFROMTEMPLATE(v, π, k, η). Therefore, per Corollary 3.2,

$$\Pr[v_k = u] \in [1, 1 + O(\eta^{1/7})] \cdot \lambda \cdot p^k_v(u)$$

for $\lambda = \frac{1}{k!}\eta^k$ independent of u and v. Now, conditioning on $v_k \neq \perp$ increases this probability by at most a factor of $1/\lambda$, as $v_k \neq \perp$ will always hold if the sampled template corresponds to a valid walk out of v_0 (which happens with probability at least $1/k!$) and every edge on that walk is in the right length-η window (which happens with probability η^k), and so $\Pr[v_k \neq \perp] \geq \lambda$. Therefore,

$$\Pr[v_k = u \mid v_k \neq \perp] \leq (1 + O(\eta^{1/7}))p_v^k(u) \leq 2d(u)/n.$$

For $j < k$, we note that v_j is distributed essentially the same as the last vertex u_j output by WalkFromTemplate(v, π, j, η). (There is one difference: it would be identical if the the $\widehat{d}$ computed in Line 17 of WalkFromTemplate used $\sigma[\eta j, 1]$ rather than $\sigma[\eta k, 1]$. But this difference has no bearing on the proof of Corollary 3.2, which just uses that this interval includes $[\eta^{99/100}, 1)$ but not $[0, \eta j]$.) So we still have that v_j is distributed within a $1 + O(\eta^{1/7})$ factor of being proportional to a true j-step random walk, and the result holds. ∎

Claim 4.7. *For $j \in [k]$ the posterior distribution of $(t_f)_{f \in E}$ given $\mathcal{F}_j$ is a product distribution. For every $f \in E \setminus E_j^*$ the distribution of t_f is uniform on $[0, 1]$.*

Proof. The only edge timestamps that influence $\mathcal{F}_j$ are those of edges in $\Psi(\vec{\mathbf{v}}_{\leq j-1})$, and so as we are conditioning on the value of all of those, and the prior distribution of $(t_f)_{f \in E}$ is a product distribution of uniform distributions, the result follows. ∎

Lemma 4.8. *For every $j \in [k]$, consider the execution of* SamplesWithResetHybrid$(k, \varepsilon, s, j-1)$. *Let H_j^i be the value of H_j in invocation i of* WalkFromTemplateHybrid. *For every $\mathcal{F}_j$, one has with probability at least $1 - n^{-2}$ over timestamps of edges in $E \setminus E_j^*$ that, for every $i \in [s]$ such that $\pi_j^i = j$ and $v_{j-1}^i \in V \setminus L$, we have that:*

$$(1 - n^{-1/9})\eta \cdot d(v_{j-1}^i) \leq |H_j^i| \leq (1 + n^{-1/9})\eta \cdot d(v_{j-1}^i),$$

and

$$(1 - n^{-1/9})\eta \cdot d(v_{j-1}^i) \leq |H_j^i \setminus E_j^*| \leq (1 + n^{-1/9})\eta \cdot d(v_{j-1}^i).$$

Finally, for every such $i \in [s]$ with $v_{j-1}^i \in V \setminus L$ and every $g \in [k]$, the degree estimate $\widehat{d}_{j-1}^g$ in invocation i satisfies

$$(1 - n^{-1/9})(1 - k\eta) \cdot d(v_{j-1}^i) \leq \widehat{d}(v_{j-1}^j) \leq (1 + n^{-1/9})(1 - k\eta) \cdot d(v_{j-1}^i).$$

Proof. For each such $u = v_{j-1}^i$, the edges in $S := \delta(u) \setminus E_j^*$ have timestamps that are independent and uniform in $[0, 1]$.

Therefore $|H_j^i \cap S|$ is distributed as the binomial variable $B(|S|, \eta)$, so by the Chernoff bound we have with $1 - \frac{1}{n^3}$ probability that

$$\big||H_j^i \cap S| - \eta|S|\big| \leq \sqrt{2\eta|S| \log 2n^3}.$$

Suppose this happens. Every remaining edge, in $E_j^* \cap \delta(u)$, is between u and another vertex collected in some walk in v; hence there are at most sk such edges.

Then by the triangle inequality:

$$\begin{aligned}
\big||H_j^i| - \eta d(u)\big| &\leq \big||H_j^i| - |H_j^i \cap S|\big| + \big||H_j^i \cap S| - \eta|S|\big| + \big|\eta d(u) - \eta|S|\big| \\
&\leq sk + \sqrt{2\eta d(u) \log 2n^3} + \eta sk \\
&\leq \eta d(u) \cdot \left(\frac{2sk}{\eta d(u)} + \sqrt{\frac{2 \log 2n^3}{\eta d(u)}} \right) \\
&\leq n^{-1/9} \eta d(u).
\end{aligned}$$

where the last step uses that $d(u) \geq n^{1/4}$, $\eta \geq n^{-1/100}$, $s \leq n^{1/30}$, $k \leq \log n$, and n is sufficiently large. The bound on $|H_j^i \setminus E_j^*| = |H_j^i \cap S|$ is the same, omitting the first of the three terms in the triangle inequality. We then union bound over $i \in [s]$.

We bound $\widehat{d}^g_{j-1}$ similarly: let a be the number of edges in S that lie in $\sigma[\eta k, 1]$, which is $B(|S|, (1-k\eta))$, and so with $1 - \frac{1}{n^3}$ probability

$$\left|a - (1-k\eta)|S|\right| \le \sqrt{2(1-k\eta)|S| \log 2n^3}.$$

Then since $\widehat{d}^g_{j-1}$ only differs from a on $E^*_j \cap \delta(u)$,

$$\left|\widehat{d}^g_{j-1} - (1-k\eta)d(u)\right| \le \sqrt{2(1-k\eta)d(u)\log 2n^3} + sk + (1-k\eta)sk \le n^{-1/9}(1-k\eta)d(u)$$

and we again union bound over i. ∎

We first analyze the vertices *collected* in each step, ignoring the rejection probability. We then include the rejection probability, to analyze the vertices *sampled* in each step.

LEMMA 4.9. *For $j \in [k]$, let $(u^i_j)_{i\in[s]}$ denote the vertices collected by* SAMPLESWITHRESETHYBRID$(k, \varepsilon, s, j-1)$ *at step j, and $(\widetilde{u}^i_j)_{i\in[s]}$ denote the vertices collected by* SAMPLESWITHRESETHYBRID(k, ε, s, j) *at step j. For every $\mathcal{F}_j$ the total variation distance between $(u^i_j)_{i\in[s]}$ and $(\widetilde{u}^i_j)_{i\in[s]}$ conditioned on $\mathcal{F}_j$ is bounded by $O(sn^{-1/9})$.*

Proof. Define

$$\mathcal{I} = \{i \in [s] : v^i_{j-1} \in V \setminus L\},$$

where for every $i \in [s]$, $v^i_{j-1} \in V \cup \{\perp\}$ is the $(j-1)^{\text{th}}$ vertex collected on the i^{th} walk, where we let $v^i_{j-1} = \perp$ if the i^{th} walk terminated before the $(j-1)^{\text{th}}$ step. Note that $\mathcal{I} \subseteq [s]$, and is quite possibly a proper subset: v^i_j could be a low degree vertex, and we may also have $v^i_j = \perp$ for some $i \in [s]$. Also note that $(v^i_{<j})_{i\in[s]}$, the set of the first $j-1$ vertices traversed by the constructed walks, is a function of $\mathcal{F}_j$ (see Definition 4.4), and in particular $\mathcal{I}$ also is.

We now modify the sampling of edges incident on high degree vertices in the invocation of SAMPLESWITHRESETHYBRID(k, ε, s, j) to avoid E^*_j, and bound the corresponding loss in total variation distance. Observe that for every choice of $\mathcal{F}_j$ for every $i \in \mathcal{I}$, $\left|\delta(v^i_{j-1}) \cap E^*_j\right| \le sk$, so the total variation distance between the uniform distribution over $\delta(v^i_{j-1})$ and the uniform distribution over $\delta(v^i_{j-1}) \setminus E^*_j$ is bounded by $sk/n^{1/4}$. Define $(x^i_j)_{i\in[s]}$ to match $\widetilde{u}^i_j$ for $i \notin \mathcal{I}$, and for $i \in \mathcal{I}$ to be sampled from $\mathcal{U}(\delta(v^i_{j-1}) \setminus E^*_j$ as opposed to $\mathcal{U}(\delta(v^i_{j-1}))$ in line 10 in the j^{th} step of SAMPLESWITHRESETHYBRID(k, ε, s, j), so $TV((\widetilde{u}^i_j)_{i\in[s]}, (x^i_{j-1})_{i\in[s]}) \le O(s^2k \cdot n^{-1/4})$.

We now perform a similar modification to the edges sampled by high degree vertices at the j^{th} step in the invocation of SAMPLESWITHRESETHYBRID$(k, \varepsilon, s, j-1)$. Let $\widetilde{H}^i_j = H^i_j \setminus E^*_j$.

By Lemma 4.8, with $1 - n^{-2}$ probability we have both

$$(1 - n^{-1/9})\eta \cdot d(v^i_{j-1}) \le |H^i_j| \le (1 + n^{-1/9})\eta \cdot d(v^i_{j-1}). \tag{4.13}$$

and

$$(1 - n^{-1/9})\eta \cdot d(v^i_{j-1}) \le |\widetilde{H}^i_j| \le (1 + n^{-1/9})\eta \cdot d(v^i_{j-1}). \tag{4.14}$$

for every $i \in \mathcal{I}$.

Conditioned on this high probability event E_H, we have using (4.13) and (4.14) that for every choice of $\mathcal{F}_j$ for every $i \in \mathcal{I}$ the total variation distance between the uniform distribution over H^i_j and the uniform distribution over $\widetilde{H}^i_j$ is bounded by $O(n^{-1/9})$. Therefore we can define $(y^i_j)_{i\in[s]}$ to match u^i_j for $i \notin \mathcal{I}$, and for $i \in \mathcal{I}$ to sample from $\mathcal{U}(\widetilde{H}^i_j)$ as opposed to $\mathcal{U}(H^i_j)$ in line 10 in the j^{th} step of SAMPLESWITHRESETHYBRID$(k, \varepsilon, s, j-1)$. This satisfies $d_{TV}((u^i_j)_{i\in[s]}, (y^i_j)_{i\in[s]}) \le O(sn^{-1/9})$.

Now, $(x^i_j)_{i\in[s]}$ and $(y^i_j)_{i\in[s]}$ are identically distributed conditioned on $\mathcal{F}_j$ and E_H. To see this, note that SAMPLESWITHRESETHYBRID$(k, \varepsilon, s, j-1)$ and SAMPLESWITHRESETHYBRID(k, ε, s, j) behave identically for $i \notin \mathcal{I}$. For $i \in \mathcal{I}$, $x^i_j \sim \mathcal{U}(\delta(v^i_{j-1}) \setminus E^*_j)$ and $y^i_j \sim \mathcal{U}(\widetilde{H}^i_j)$, both independently of the algorithms' behavior on $i' \ne i$.

Since $\widetilde{H}_j^i$ is a random binomial sample of $\delta(v_{j-1}^i) \setminus E_j^*$, y_j^i is also uniform over $\delta(v_{j-1}^i) \setminus E_j^*$ conditioned on $|\widetilde{H}_j^i|$, as long as $\widetilde{H}_j^i \neq \emptyset$; thus it is conditioned on E_H.

As a result, the outputs of the unmodified calls to SamplesWithResetHybrid$(k, \varepsilon, s, j-1)$ and SamplesWithResetHybrid(k, ε, s, j) are $O(s^2 k n^{-1/4} + n^{-2} + s n^{-1/9}) = O(s n^{-1/9})$ close in total variation distance. ∎

Lemma 4.10. *For $j \in [k]$, if $(v_j^i)_{i\in[s]}$ denote the vertices sampled by* SamplesWithResetHybrid$(k, \varepsilon, s, j-1)$ *at step j, and $(\widetilde{v}_j^i)_{i\in[s]}$ denote the vertices sampled by* SamplesWithResetHybrid(k, ε, s, j) *at step j, then for every $\mathcal{F}_j$ the total variation distance between $(v_j^i)_{i\in[s]}$ and $(\widetilde{v}_j^i)_{i\in[s]}$ is bounded by $O(s n^{-1/9})$.*

Proof. By Lemma 4.9, the distribution of vertices *collected* in step j is $O(s n^{-1/9})$ close in total variation distance. Therefore it suffices to show that the rejection sampling is similarly close.

We condition on $\mathcal{F}_j$. Let $(u_j^i)_{i\in[s]}$ denote the vertices collected by SamplesWithResetHybrid$(k, \varepsilon, s, j-1)$ at step j, and $(\widetilde{u}_j^i)_{i\in[s]}$ denote the vertices collected by SamplesWithResetHybrid(k, ε, s, j) at step j. By Lemma 4.9 the total variation distance between $(u_j^i)_{i\in[s]}$ and $(\widetilde{u}_j^i)_{i\in[s]}$ is bounded by $O(s n^{-1/9})$. In what follows we analyze the rejection sampling step in line 24 of WalkFromTemplateHybrid. We show that the fact that the degrees of vertices in $V \setminus L$ are estimated using their actual degrees (in line 21 of WalkFromTemplateHybrid) as opposed to using the stream (in line 17 of WalkFromTemplate) leads to only $O(s \cdot n^{-1/9})$ contribution to total variation distance.

We note that degree estimates computed for vertices in L (i.e., low degree vertices) are the same in both invocations of SamplesWithResetHybrid, and consider vertices in $V \setminus L$, i.e. high degree vertices. Let $u = u_{j-1}^i$ for some $i \in [s]$, and suppose that $u \in V \setminus L$.

Define

$$\lambda = \prod_{g=1}^{j-1} \frac{\min\left(\frac{\eta}{\gamma_g \widehat{d}_{g-1}^{j}}, 1\right)}{\min\left(\frac{\eta}{\gamma_g \widehat{d}_{g-1}^{j-1}}, 1\right)}.$$

Since $\widehat{d}_{g-1}^{j-1} \leq \widehat{d}_{g-1}^{j}$, $\lambda \leq 1$. The probability that u is accepted by the rejection sampling is

$$\frac{\alpha_j}{\alpha_{j-1}} = \lambda \min\left(\frac{\eta}{\gamma_j \widehat{d}_{j-1}^{j}}, 1\right).$$

The only difference in the rejection sampling between the two executions is that γ_j and $\widehat{d}_{j-1}^{j}$ are different. Let γ_j, $\widetilde{d}_{j-1}^{j}$ be the values in the $j^* = j-1$ case, and $\widetilde{\gamma}_j = \frac{1}{d(u)}$, $\widetilde{\widetilde{d}}_{j-1}^{j} = (1-k\eta)d(u)$ be the values these take in the $j^* = j$ case. The total variation incurred by this change in rejection sampling probability is thus

$$\begin{aligned}
\delta &\leq \left| \lambda \min\left(\frac{\eta}{\gamma_j \widehat{d}_{j-1}^{j}}, 1\right) - \lambda \min\left(\frac{\eta}{\widetilde{\gamma}_j \widetilde{\widetilde{d}}_{j-1}^{j}}, 1\right) \right| \\
&\leq \lambda \left| \frac{\eta}{\gamma_j \widehat{d}_{j-1}^{j}} - \frac{\eta}{(1-\eta k)} \right| \\
&\leq \left| \frac{|H_j|}{\widehat{d}_{j-1}^{j}} - \frac{\eta}{(1-\eta k)} \right|.
\end{aligned}$$

Now, by Lemma 4.8, conditioned on $\mathcal{F}_j$ with at least $1 - n^{-2}$ probability we have

$$|H_j| \in (1 \pm n^{-1/9})\eta d(u)$$

and

$$\widehat{d}^{j}_{j-1} \in (1 \pm n^{-1/9})(1 - k\eta)d(u)$$

so that

$$\delta \leq \frac{\eta}{(1-\eta k)} \cdot \left(\frac{1 + n^{-1/9}}{1 - n^{-1/9}} - 1 \right) = O(\eta / n^{1/9}).$$

The net result is that every high degree vertex u has a rejection sampling step that is $O(n^{-1/9})$ close in the two cases. This means that all s rejection sampling steps are $O(sn^{-1/9})$ close in the two cases. Combining with with the collection being close (Lemma 4.9) gives the result. ∎

Proof of Lemma 4.1: We use induction on Lemma 4.10. The result follows by noting that SamplesWithReset(k, ε, s) is the same as SamplesWithResetHybrid$(k, \varepsilon, s, 0)$ and applying triangle inequality for total variation distance, to get a total variation distance of $O(skn^{-1/9}) \leq O(sn^{-1/10})$ for $k = O(\log n)$. □

4.3 Near-Independence of Hybrid Algorithm To establish the near-independence of our walks, we compare the output distribution of SamplesWithResetHybrid(k, ε, s, k) run on a random order stream σ to the output of an auxiliary version of SamplesWithResetHybrid(k, ε, s, k), in which the invocations of WalkFromTemplateHybrid$(v_0^i, \pi^i, k, \eta, k)$ are run on independent streams $\widetilde{\sigma}^i$.

Let $\vec{\mathbf{v}}$ denote the s partial walks sampled (see Definition 4.2) by WalkFromTemplateHybrid$(v_0^i, \pi^i, k, \eta, k)$ on stream σ, and let $\vec{\mathbf{w}}$ denote the s partial walks that would be sampled if the s invocations each used their own independent random streams $(\widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n)$.

Definition 4.11. *We say a collection of (partial) walks $\vec{\mathbf{w}}$ is "well-separated" if for all $\mathbf{w}, \mathbf{w}' \in \vec{\mathbf{w}}$ we have $\Psi(\mathbf{w}) \cap \Psi(\mathbf{w}') = \emptyset$.*

Well-separatedness is equivalent to the conditions:

- No vertex in L is visited in more than one walk.
- No pair of adjacent vertices in L is visited in more than one walk.

Lemma 4.12. *$\vec{\mathbf{w}}$ is well-separated with $1 - O(k \cdot n^{-1/11})$ probability.*

Proof. By induction on Lemma 4.10, the distribution of walks generated by SamplesWithResetHybrid$(k, \varepsilon, 1, k)$ and SamplesWithResetHybrid$(k, \varepsilon, 1, 0)$ are $O(kn^{-1/9})$ close in total variation distance. Therefore it suffices to show this lemma for for $\vec{\mathbf{w}}$ drawn from s calls to WalkFromTemplateHybrid$(v_0^i, \pi^i, k, \eta, 0)$ on independent random streams, i.e. $j^* = 0$ rather than $j^* = k$.

Let $p_j(u) = \Pr[\mathbf{w}_j^i = u]$ be the marginal probability of sampling vertex u in step j, which is independent and identically distributed over $i \in [s]$. By Lemma 4.6, $p_j(u) \leq d(u)/n$ for all u and j.

Consider any $j, j' \in [k]$ and distinct $i, i' \in [s]$. For any fixed $u \in L$, the probability that $w_j^i = u = w_{j'}^{i'}$ is $p_j(u)^2 \leq d(u)^2/n^2 \leq 1/n^{3/2}$. Taking a union bound over $(sk)^2$ choices of (i, j, i', j') and n of u, the chance that any vertex in L is visited in more than one walk is at most $(sk)^2/n^{1/2}$.

For any fixed edge $(u, v) \in (L \times L) \cap E$, the probability that $w_j^i = u$ and $w_{j'}^{i'} = v$ is at most $p_j(u)p_j(v) \leq 1/n^{3/2}$. Taking a union bound over the $n \cdot n^{1/4}$ total edges out of vertices in L, the chance this ever happens is at most $(sk)^2/n^{1/4}$. ∎

When $\vec{\mathbf{w}}$ is well-separated, the edges whose timestamps are looked at by WalkFromTemplateHybrid$(v_0^i, \pi^i, k, \eta, k)$ are different across each invocation $i \in [s]$. This lets us couple the independent streams case to the dependent streams case.

LEMMA 4.13. *Assuming* ***(P1)***, ***(P2)*** *and* $s \le n^{1/30}$, *for every input graph* G

$$d_{TV}(\vec{\mathbf{w}}, \vec{\mathbf{v}}) \le n^{-1/12}$$

Proof. We partition the internal randomness used by SAMPLESWITHRESETHYBRID into s disjoint independent random strings $R^1, \ldots, R^s$, such that for every $i \in [s]$ the i^{th} invocation of WALKFROMTEMPLATEHYBRID uses string R^i.

Note that the behavior of the i^{th} invocation of WALKFROMTEMPLATEHYBRID is fully determined by its randomness R^i and by the timestamps of edges in $\Psi(\mathbf{w}^i)$ (the low-degree vertices; see Definition 4.3). Given these timestamps, WALKFROMTEMPLATEHYBRID is invariant under other changes to the stream.

Let $\mathcal{E}$ denote the event that the independent walk case $\vec{\mathbf{w}}$ is "well-separated", meaning the $\Psi(\mathbf{w}^i)$ do not overlap. Per Lemma 4.12, $\mathcal{E}$ holds with $1-O(kn^{-1/11})$ probability. When $\mathcal{E}$ holds, we can "couple" the independent stream result $\vec{\mathbf{w}}$ to the single stream result $\vec{\mathbf{v}}$ because each element $\vec{\mathbf{v}}^i$ depends on disjoint edges in the stream. That is, we use the timestamps used by the independent walk algorithms to construct a single stream that is both correctly distributed (all of its timestamps uniform and independent) such that, if every instance of WALKFROMTEMPLATEHYBRID was run on this stream, their executions would be identical to the executions of the original independent copies of WALKFROMTEMPLATEHYBRID whenever $\mathcal{E}$ holds.

We now give the details of coupling $\vec{\mathbf{w}}$ and $\vec{\mathbf{v}}$ under $\mathcal{E}$. We construct a distribution $\mathcal{D}$ of a single stream and internal randomness (σ, R) from $(\widetilde{\sigma}^i, R^i)_{i\in[s]})$, such that the marginal distribution of (σ, R) is that of the stream and randomness underlying $\vec{\mathbf{v}}$, and so that, if the stream and randomness underling $\vec{\mathbf{v}}$ is set to be (σ, R), $\vec{\mathbf{w}} = \vec{\mathbf{v}}$ whenever $\mathcal{E}$ holds.

Given the independent streams $(\widetilde{\sigma}^i)_{i\in[s]}$, we can construct the dependent instance (σ, R) as follows. We set $R = \widetilde{R}$, and just need to fix the timestamps of edges in σ. For each edge $f \in E$, we say that an edge $f \in E$ is *covered* by $i \in [s]$ if $f \in \Psi(\mathbf{w}^i)$. For $f \in E$, let t_f be the timestamp of f in σ and $\widetilde{t}^i_f$ be the timestamp of f in $\widetilde{\sigma}^i$. We set

$$t_f = \begin{cases} \widetilde{t}^i_f & \text{if } i \text{ is the smallest that covers } f \\ \mathcal{U}[0,1] & \text{if } f \text{ is not covered by any } i \in [s]. \end{cases} \tag{4.15}$$

Note that the resulting distribution of $(t_f)_{f\in E}$ is indeed a product of uniform distributions over $[0,1]$. Indeed, one can think of $(t_f)_{f\in E}$ using the principle of deferred decisions: starting with v^1_0, we run WALKFROMTEMPLATEHYBRID from v^1_0, sampling timestamps of edges as soon as they are needed, and then proceeding for $v^2_0, v^3_0, \ldots, v^s_0$. The first time the edge is covered by a walk, its timestamp is sampled from the uniform distribution, independently of the other timestamps, as required.

It remains to note that conditioned on $\mathcal{E}$ for every $f \in E$ there exists at most one $i \in [s]$ such that f is covered by i. Therefore, when $\mathcal{E}$ holds, the timestamps of $\Psi(\mathbf{w}^i)$ are the same in σ and $\widetilde{\sigma}^i$. Since the randomness is also the same, this means $\mathbf{v}^i = \mathbf{w}^i$ for each i, or $\vec{\mathbf{v}} = \vec{\mathbf{w}}$. Since $\mathcal{E}$ occurs with probability $1 - kn^{-1/11}$, we have that the two outputs match with probability $1 - kn^{-1/11} > 1 - n^{-1/12}$, as desired. ∎

5 Proof of Theorem 1.3

We now give

Proof of Theorem 1.3:

Correctness. Let $S = (v^1_0, \ldots, v^s_0)$, where $v^i_0 \sim \mathcal{U}(V)$ are chosen uniformly at random with replacement. We consider four distribution on walks in our proof, which we define below.

Walks generated by SAMPLESWITHRESET **on independent streams.** Let $\sigma_1, \ldots, \sigma_s$ be independent random order streams. Let $\vec{\mathbf{v}}_* = (\mathbf{v}^i_*)_{i\in[s]}$ denote walks generated by SAMPLESWITHRESET(k, ε, s) (Algorithm 3), where the i^{th} walk is generated on σ^i.

Walks generated by SAMPLESWITHRESETHYBRID **on independent streams.** Let $\sigma_1, \ldots, \sigma_s$ be independent random order streams. Let $\vec{\mathbf{x}} = (\mathbf{x}^i)_{i\in[s]}$ denote walks generated by SAMPLESWITHRESETHYBRID(k, ε, s, k) (Algorithm 6), where the i^{th} walk is generated on σ^i.

Walks generated by SamplesWithResetHybrid **on a joint stream.** Let $\vec{\mathbf{y}} = (\mathbf{y}^i)_{i\in[s]}$ denote walks generated by SamplesWithResetHybrid(k, ε, s, k) (Algorithm 6), where every walk is generate on the same stream σ.

Walks generated by SamplesWithReset **on a joint stream.** Let $\vec{\mathbf{v}} = (\mathbf{v}^i)_{i\in[s]}$ denote walks generated by SamplesWithReset(k, ε, s) (Algorithm 3), where every walk is generate on the same stream σ.

Proof outline. Note that $\vec{\mathbf{v}}$ is the random variable that we obtain from our random order streaming algorithm SamplesWithReset. We first show that it is close in distribution to $\vec{\mathbf{v}}_*$, and then show that the distribution of $\vec{\mathbf{v}}_*$ is such that SimulateWalks has the desired property.

Step 1: showing that $\vec{\mathbf{v}}_* \approx \vec{\mathbf{v}}$. By Lemma 4.1,

$$d_{TV}(\vec{\mathbf{v}}, \vec{\mathbf{y}}) \le O(s \cdot n^{-1/10})$$

Second, by Lemma 4.13,

$$d_{TV}(\vec{\mathbf{y}}, \vec{\mathbf{x}}) \le n^{-1/12}.$$

Third, by Lemma 4.1 invoked with $s = 1$ applied to each independent stream, together with triangle inequality for total variation distance, we have

$$d_{TV}(\vec{\mathbf{x}}, \vec{\mathbf{v}}_*) \le \sum_{i=1}^{s} d_{TV}(\mathbf{x}^i, \mathbf{v}_*^i) \le O(s \cdot n^{-1/10}).$$

Indeed, running SamplesWithReset (respectively SamplesWithResetHybrid) with every internal invocation of WalkFromTemplate using a separate stream is equivalent to a concatenation of independent instances of SamplesWithReset (respectively SamplesWithResetHybrid) with $s = 1$. Combining these three equations, the triangle inequality gives

$$d_{TV}(\vec{\mathbf{v}}, \vec{\mathbf{v}}_*) \le O(sn^{-1/10}) + n^{-1/12} < n^{-1/100}$$

for sufficiently large n, which it will be when c' is sufficiently small, as $\sqrt{c' \log n} \ge 1$.

Step 2: verifying preconditions of Lemma 4.1 and Lemma 4.13. We now verify that **(P1)** and **(P2)** hold. Let the constants $c, C > 0$ be the ones chosen in Lemma 3.1. First, **(P1)** is satisfied since

$$k \le \sqrt{c' \log n} \le c \log n / \log\log n$$

for a sufficiently small constant $c' > 0$ by assumption. The upper bound in **(P2)** is satisfied since

$$\eta = \varepsilon^8 \cdot 2^{-Ck} \le 2^{-Ck}$$

by the choice of η in Algorithm 3. The lower bound in **(P2)** is satisfied since

$$\eta = \varepsilon^8 \cdot 2^{-Ck} \ge n^{-1/101} \cdot 2^{-Ck} = n^{-1/101 - o(1)} \ge n^{-1/100},$$

since $\varepsilon \ge n^{-1/1000}$ and $2^{Ck} = 2^{O(\sqrt{\log n})} = n^{o(1)}$ by the assumption on k. Furthermore, we have

$$s = b \cdot (1/\varepsilon)^{O(k)} \cdot 2^{O(k^2)} \le n^{1/100} \cdot 2^{O(c' \log n)} \le n^{1/30}$$

since $b \le n^{1/100}$ by assumption of the theorem and

$$k \le \min\left\{ \frac{c' \log n}{\log(1/\varepsilon)}, \sqrt{c' \log n} \right\}$$

for a sufficiently small $c' > 0$ by assumption of the theorem.

Step 3: showing that $\vec{\mathbf{v}}_*$ leads to the required distribution. The output of SimulateWalks (Algorithm 4) is the first b successful walks from SamplesWithReset, which is (per step 1) $n^{-1/100}$-close to the first b successful walks from $\vec{\mathbf{v}}_*$, for $s = b \cdot 100\eta^{-k} \cdot k!$.

For any fixed k-step walk $\mathbf{w}$, let

$$p_*(\mathbf{w}) := \frac{1}{n} \prod_{i=0}^{k-1} \frac{1}{d(w_i)}$$

be the true probability that a k-step random walk from a uniform vertex equals $\mathbf{w}$. By taking the average of Corollary 3.2 over uniform initial vertices v, we have for any $i \in [s]$ that

$$\Pr[\mathbf{v}_*^i = \mathbf{w}] \in [1, 1 + O(\eta^{1/7})] \cdot \frac{\eta^k}{k!} \cdot p_*(\mathbf{w}).$$

As a result,

$$\Pr[\mathbf{v}_*^i = \mathbf{w} \mid \mathbf{v}_*^i \text{ succeeds}] \in [1, 1 + O(\eta^{1/7})] \cdot p_*(\mathbf{w}) \tag{5.16}$$

and

$$\Pr[\mathbf{v}_*^i \text{ succeeds}] \geq \frac{\eta^k}{k!}.$$

By construction, the $\mathbf{v}_*^i$ are independent across $i \in [s]$. We expect at least $\frac{\eta^k}{k!}s = 100b$ repetitions to succeed, so by a Chernoff bound at least a will succeed with at least $1-2^{-b}$ probability. Therefore the output of SimulateWalks is $(n^{-1/100} + 2^{-b})$-close to the distribution of a independent samples from $(\mathbf{v}_*^i = \mathbf{w} \mid \mathbf{v}_*^i \text{ succeeds})$. By (5.16), each such sample is $O(\eta^{1/7}) < \varepsilon$-close to a uniform random walk.

Step 4: space complexity. The expected space complexity of a single invocation of WalkFromTemplate (Algorithm 5) is $O(k)$, so the overall space complexity is $(1/\varepsilon)^{O(k)}2^{O(k^2)}b$, as required. □

6 Digraph Lower Bound

In this section, we prove that both sampling random walks and approximating the PageRank of a given vertex set are hard in *directed* graph streams.

Theorem 6.1. *For any constant $\varepsilon < 1/4$, the following holds for all $k \geq 3$ and all n: there is a family of directed graphs with no more than n vertices and edges such that any random order streaming algorithm that ε-approximates the distribution of length-k random walks on graphs drawn from the family uses $\Omega(n)$ bits of space, with a constant factor depending only on ε.*

Theorem 6.2. *Let α be a given (constant) reset probability for PageRank. For any constants $\varepsilon < (1-\alpha)^3 - \frac{1}{2}$, $\delta < 1/4$, the following holds for all n: there is a family of directed graphs with no more than n vertices and edges such that any random order streaming algorithm that returns a ε additive approximation to the PageRank of vertex sets in these graphs with probability $1-\delta$ uses $\Omega(n)$ bits of space, with a constant factor depending only on α, ε, and δ.*

We will prove these by a reduction from INDEX_n. In this one-way communication problem Alice has a string $x \in \{0,1\}^n$ while Bob has an index $I \in [n]$. Alice must send Bob a message such that he can determine x_I. We will show that a *uniform* instance of this problem can be converted into a random graph stream such that approximating the random walk distribution or the PageRank vector allows solving indexing.

The following is a well-known consequence of information theory—for completeness, we include a proof in Appendix D.

Lemma 6.3. *Let $\varepsilon > 0$ be any constant. Any protocol that solves INDEX_n on a uniform input with probability $1/2 + \varepsilon$ requires $\Omega(n)$ communication in expectation.*

6.1 Graph Distribution We start by defining a distribution on length-O(n) directed graph streams that Alice and Bob can construct (with a prefix belonging to Alice and a postfix belonging to Bob, and interleaved edges that they construct using shared randomness that are independent of their input) using their inputs to INDEX$_n$.

We will show that the graph streams correspond to randomly choosing a graph and then uniformly permuting its edges, and any algorithm that generates a distribution that is ε-close to either the random walk distribution or the PageRank distribution for some constant $\varepsilon < 1/4$ will be able to use this to solve INDEX$_n$.

Vertices. There will be βn vertices $(a_i)_{i=1}^{\beta n}$ where $\beta \in \mathbb{N}_{>0}$ is a constant depending on ε, a single vertex b, n vertices $(c_i)_{i=1}^{n}$, and two pairs of vertices $\{d_0, e_0\}$, $\{d_1, e_1\}$. Every vertex in the graph will have a path to either $\{d_0, e_0\}$ or $\{d_1, e_1\}$, which will function as "sink" sets.

Fixed Edges. These edges will not depend on either player's input. The players can use shared randomness to insert them uniformly at random among their other edges.

They are $(a_i b)_{i=1}^{\beta n}$, i.e. a star of βn edges pointing into b and the four edges d_0e_0, e_0d_0, d_1e_1, and e_1d_1 (i.e. meaning that each of $\{d_0, e_0\}$ and $\{d_1, e_1\}$ is a 2-vertex loop).

Alice's edges. Let π be a uniformly random permutation of $[n]$, and let J be drawn uniformly from $\{0, \dots, n\}$. These will be used to define the boundary between Bob's edges and Alice's edges.

Recall that Alice's input is a string $x \in \{0,1\}^n$. For each $i \in [J]$, Alice has the edge $c_{\pi(i)}d_{x_{\pi(i)}}$, with $c_{\pi(1)}d_{x_{\pi(1)}}$ first, $c_{\pi(2)}d_{x_{\pi(2)}}$ second, and so on.

Bob's edges Recall that Bob has the index I. His first edge will be bc_I. Then, for each $i \in \{J+1, \dots n\}$, he has the edge $c_{\pi(i)}d_{y_{\pi(i)}}$, where y is a random n-bit string.

LEMMA 6.4. *The graph stream described above is a uniformly random order graph stream.*

Proof. Fix any value of Bob's index I. If we were to randomly draw a string $z \in \{0,1\}^n$, randomly order edges $(c_i d_{z_i})_{i=1}^{n}$, and then insert bc_I and then our fixed edges randomly in this stream, this would give a uniformly random order graph stream (corresponding to drawing a graph from a fixed distribution and then randomly permuting its edges).

But this would also be identically distributed to our graph stream, as the strings x and y are both drawn uniformly at random from $\{0,1\}^n$. So we have a uniformly random graph stream. ∎

LEMMA 6.5. *With probability* 3/4*, every vertex in* $(a_i)_{i=1}^{\beta n}$ *has a length-*3 *path from it to* $\{d_{x_I}, e_{x_I}\}$*, and (up to prefixes) that is the only path out of that vertex.*

Proof. Each has an edge to b, which has a single edge to c_J. With probability 1/2, $\pi(I) \le J$, and so the unique edge out of c_J points to d_{x_I}. Otherwise, it points to d_{y_I}, which is d_{x_I} with probability 1/2, as y is drawn at random. ∎

We will now prove that both the random walk distribution and the PageRank vector on this graph can be used to determine the answer to INDEX$_n$. Note that there are no vertices with out-degree 0 in this graph, so we do not need to concern ourselves with what a random walk or the PageRank walk should do when encountering such a vertex.

6.2 Random Walks

LEMMA 6.6. *Let* $\mathcal{A}$ *be any algorithm using* S *space such that, when given a sample from the distribution on graph streams above, the output of* $\mathcal{A}$ *is* ε*-close in total variation distance to sampling a* k*-step random walk from a random vertex in the graph given by the stream, where* $k \ge 3$*. Then there is a protocol for* INDEX$_n$ *on uniform inputs that uses* S *space and succeeds with probability* $3/4 - \varepsilon - 6/\beta$*.*

Proof. The protocol will be as follows:

1. Alice and Bob use their INDEX input to construct a corresponding random graph stream.
2. Using S bits of communication from Alice to Bob, they run $\mathcal{A}$ on the stream.
3. If the walk output by $\mathcal{A}$ ends in $\{d_z, e_z\}$ for $z \in \{0,1\}$, Bob answers z. Otherwise he answers arbitrarily.

Now, by Lemma 6.5, with probability 3/4 any length-k random walk starting from a vertex in $(a_i)_{i=1}^{\beta n}$ will reach $\{d_{x_I}, e_{x_I}\}$, and as that set has no out-edges it will stay there. The probability of starting at one of these vertices is

$$\begin{aligned}\frac{\beta n}{\beta n + 1 + n + 4} &= \frac{1}{1 + 1/\beta + 5/n\beta} \\ &= \frac{1}{1 + 6/\beta} \\ &\leq 1 - 6/\beta\end{aligned}$$

and so the probability that the distribution given by $\mathcal{A}$ ends in $\{d_{x_I}, e_{x_I}\}$ is at least

$$3/4 - \varepsilon - 6/\beta$$

proving the correctness of the protocol. The construction of the stream itself is done entirely with public randomness, so this gives an S bit public randomness protocol. As we are working with a fixed input distribution, this means that there is also an S bit private randomness protocol, by fixing whichever set of public random bits maximizes the probability of success over the uniform distribution. □

We are now ready to prove Theorem 1.7.

THEOREM 6.7. *For any constant $\varepsilon < 1/4$, the following holds for all $k \geq 3$ and all n: there is a family of directed graphs with no more than n vertices and edges such that any random order streaming algorithm that ε-approximates the distribution of length-k random walks on graphs drawn from the family uses $\Omega(n)$ bits of space, with a constant factor depending only on ε.*

Proof. Set $\beta = \frac{12}{1/4 - \varepsilon}$. Then, for each n, we can construct the family given by the stream distribution described in section 6.1 from $\text{INDEX}_{n'}$, where $n' = \Omega(n)$ while keeping the total number of vertices and edges below n. By Lemma 6.6, any algorithm that ε-approximates the distribution of length-k random walks on graphs drawn from this family gives a protocol for $\text{INDEX}_{n'}$ that succeeds with probability

$$3/4 - \varepsilon - 6/\beta = 1/2 + \frac{1/4 - \varepsilon}{2}$$

and so by Lemma 6.3 it uses $\Omega(n') = \Omega(n)$ space, where the constant depends only on ε and β. So as β depends only on ε the result follows. □

6.3 PageRank

LEMMA 6.8. *With probability at least 3/4 over the construction of the graph stream, the PageRank vector with reset probability α has support at least $(1 - 6/\beta)(1 - \alpha)^3$ on $\{d_{x_I}, e_{x_I}\}$.*

Proof. Recall that the PageRank vector is the stationary distribution of the Markov chain in which each step is a random walk step with probability $(1 - \alpha)$ and a jump to a uniformly random vertex with probability α. By Lemma 6.5, with probability 3/4 over the graph generation process, every length-3 or greater walk from a vertex in $(a_i)_{i=1}^{\beta n}$ reaches $\{d_{x_I}, e_{x_I}\}$, and as this set has no out-edges, it stays there.

Suppose this holds. Then starting from any point, after $k \geq 3$ steps on the chain, sufficient criteria to be in $\{d_{x_I}, e_{x_I}\}$ are that

(1) There has been at least one jump, but not in the last 3 steps.

(2) That jump went to a vector in $(a_i)_{i=1}^{\beta n}$.

Criterion **(1)** will hold with probability at least

$$(1-(1-\alpha)^{k-3})(1-\alpha)^3$$

which converges to $(1-\alpha)^3$ as $k \to \infty$. Conditioned on this, criterion **(2)** will hold with probability $\frac{\beta n}{\beta n+1+n+2} \geq 1-6/\beta$. So as $k \to \infty$, the probability that a walk on the chain is in $\{d_{x_I}, e_{x_I}\}$ converges to at least

$$(1-6/\beta)(1-\alpha)^3$$

completing the proof. ∎

LEMMA 6.9. *Let $\mathcal{A}$ be any algorithm using S space such that, when given a sample from the distribution on graph streams above, it with probability $1-\delta$ outputs an ε-additive approximation to the PageRank of the set $\{d_0, e_0\}$ with reset probability α, where $\varepsilon < (1-6/\beta)(1-\alpha)^3 - 1/2$. Then there is a protocol for* INDEX_n *on uniform inputs that uses S space and succeeds with probability $3/4-\delta$.*

Proof. The protocol will be as follows:

1. Alice and Bob use their INDEX input to construct a corresponding random graph stream.
2. Using S bits of communication from Alice to Bob, they run $\mathcal{A}$ on the stream, estimating the PageRank of $\{d_0, e_0\}$.
3. If the sample output by $\mathcal{A}$ is at least $1/2$, Bob outputs 0. Otherwise he outputs 1.

Now, by Lemma 6.8, with probability $3/4$, the PageRank vector of the graph has support at least

$$(1-6/\beta)(1-\alpha)^3$$

on $\{d_{x_I}, e_{x_I}\}$ and so with probability $3/4-\delta$, the algorithm will report the correct answer.

The construction of the stream itself is done entirely with public randomness, so this gives an S bit public randomness protocol. As we are working with a fixed input distribution, this means that there is also an S bit private randomness protocol, by fixing whichever set of public random bits maximizes the probability of success over the uniform distribution. ∎

We are now ready to prove Theorem 1.8.

THEOREM 6.10. *Let α be a given (constant) reset probability for PageRank. For any constants $\varepsilon < (1-\alpha)^3 - \frac{1}{2}$, $\delta < 1/4$, the following holds for all n: there is a family of directed graphs with no more than n vertices and edges such that any random order streaming algorithm that returns a ε additive approximation to the PageRank of vertex sets in these graphs with probability $1-\delta$ uses $\Omega(n)$ bits of space, with a constant factor depending only on α, ε, and δ.*

Proof. Set $\beta = \frac{12}{(1-\alpha)^3 - \frac{1}{2} - \varepsilon}$. Then, for each n, we can construct the family given by the stream distribution described in section 6.1 from $\text{INDEX}_{n'}$, where $n' = \Omega(n)$ while keeping the total number of vertices and edges below n. By Lemma 6.9, as

$$(1-6/\beta)(1-\alpha)^3 - 1/2 \geq (1-\alpha)^3 - 6/\beta - 1/2 \geq \frac{1}{2}((1-\alpha)^3 - 1/2 + \varepsilon) > \varepsilon$$

this gives a protocol for $\text{INDEX}_{n'}$ that succeeds with probability

$$3/4 - \delta > 1/2$$

and so by Lemma 6.3 it uses $\Omega(n') = \Omega(n)$ space, where the constant depends only on α, ε, and β. So as β depends only on α and ε the result follows. ∎

Acknowledgements

Michael Kapralov was supported by ERC Starting Grant 759471.

John Kallaugher and Eric Price were supported by NSF Award CCF-1751040 (CAREER).

John was also supported by Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. Also supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing program.

References

[ACL06] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Local graph partitioning using pagerank vectors. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 475–486. IEEE Computer Society, 2006.

[AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 20–29. ACM, 1996.

[AP09] Reid Andersen and Yuval Peres. Finding sparse cuts locally using evolving sets. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 235–244. ACM, 2009.

[BLM13] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[CCM16] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. *Theory of Computing*, 12(10):1–35, 2016.

[CJMM17] Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 29:1–29:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[CKKP21] Ashish Chiplunkar, John Kallaugher, Michael Kapralov, and Eric Price. Approximating local graph structure in almost random order streams, 2021.

[CKP+21] Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Near-optimal two-pass streaming algorithm for sampling random walks over directed graphs. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 52:1–52:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[COP03] Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 30–39. ACM, 2003.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley series in telecommunications. Wiley, New York, 1991.

[DN03] Herbert A. David and Haikady N. Nagaraja. *Order Statistics*. John Wiley, Hoboken, N.J, 3rd ed. edition, 2003.

[FKM+04] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 531–543. Springer, 2004.

[Jin19] Ce Jin. Simulating random walks on graphs in the streaming model. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 46:1–46:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[KKS14] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751. SIAM, 2014.

[KMNT20] Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1753–1772. SIAM, 2020.

[MMPS17] Morteza Monemizadeh, S. Muthukrishnan, Pan Peng, and Christian Sohler. Testable bounded degree graph properties are random order streamable. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 131:1–131:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[PS18] Pan Peng and Christian Sohler. Estimating graph parameters from random order streams. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, page 2449–2466, USA, 2018. Society for Industrial and Applied Mathematics.

[SGP11] Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. Estimating pagerank on graph streams. *J. ACM*, 58(3):13:1–13:19, 2011.

[ST13] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM J. Comput.*, 42(1):1–26, 2013.

A Omitted Proofs of Technical Results

CLAIM A.1. *Let $v \in V$ be chosen uniformly at random in a graph with no isolated vertices. Then for every $u \in V$ and $k \geq 0$ one has $\mathbb{E}_{v \sim \mathcal{U}(V)}[p_v^k(u)] \leq d(u)/n$.*

Proof. The base case is given by $k = 0$: we have $\mathbb{E}_{v \sim \mathcal{U}(V)}[p_v^k(u)] = 1/n \leq d(u)/n$. The inductive step is given by:

$$\begin{aligned}
\mathop{\mathbb{E}}_{v \sim \mathcal{U}(V)}\left[p_v^{k+1}(u)\right] &= \sum_{w \in \delta(u)} \frac{1}{d(w)} \mathop{\mathbb{E}}_{v \sim \mathcal{U}(v)}\left[p_v^k(w)\right] \\
&\leq \sum_{w \in \delta(u)} \frac{1}{d(w)} \cdot \frac{d(w)}{n} \\
&= \frac{d(u)}{n}.
\end{aligned}$$

□

We will require Bennett's inequality.

THEOREM A.2. (BENNETT'S INEQUALITY, THEOREM 2.9 IN [BLM13]) *Let $X_1, \ldots, X_n$ be independent random variables with finite variance such that $X_i \leq b$ for some $b > 0$ almost surely for all $i \in [n]$. Let*

$$S = \sum_{i \in [n]} (X_i - \mathbb{E}[X_i])$$

and let $v = \sum_{i \in [n]} \mathbb{E}[X_i^2]$. Then for any $t > 0$

$$\Pr[S \geq t] \leq \exp\left(-\frac{v}{b^2} h\left(\frac{bt}{v}\right)\right),$$

where $h(u) = (1+u)\ln(1+u) - u$ for $u > 0$.

In applying Bennett's inequality, we will need the following two properties of h:

CLAIM A.3. *The function $v \cdot h\left(\frac{1}{v}\right)$ is monotone decreasing in v, where $h(u) = (1+u)\ln(1+u) - u$ for $u > 0$.*

Proof.

$$\begin{aligned}
\frac{d}{dv}\left(v \cdot h\left(\frac{1}{v}\right)\right) &= \frac{d}{dv}\left(v \cdot \left[\left(1 + \frac{1}{v}\right)\ln\left(1 + \frac{1}{v}\right) - \frac{1}{v}\right]\right) \\
&= \frac{d}{dv}\left((v+1)\ln\left(1 + \frac{1}{v}\right) - 1\right) \\
&= \frac{d}{dv}\left((v+1)(\ln(v+1) - \ln v) - 1\right) \\
&= \ln(v+1) - \ln v + 1 - 1 - 1/v \\
&= \ln(1 + 1/v) - 1/v \\
&\leq 0
\end{aligned}$$

since $\ln(1+x) \leq x$ for all $x \in (-1, +\infty)$. □

CLAIM A.4. *For every $u \geq 0$ one has $h(u) = (1+u)\ln(1+u) - u \geq \frac{1}{2} u \ln u$.*

Proof. When $u = 0$ both sides are equal, so it will suffice to verify that

$$\frac{d}{du}\left((1+u)\ln(1+u) - u - \frac{1}{2}u\ln u\right) = -\frac{1}{2} - \frac{1}{2}\ln u + \ln(1+u),$$

is nonnegative for all $u \geq 0$. The latter claim can be verified by observing that the function on the rhs above goes to $+\infty$ as $u \to 0+$ and as $u \to +\infty$, and its derivative

$$\frac{d}{du}\left(-\frac{1}{2} - \frac{1}{2}\ln u + \ln(1+u)\right) = -\frac{1}{2u} + \frac{1}{1+u}$$

has exactly one zero at $u = 1$, where $h(u)$ is positive: $-\frac{1}{2} - \frac{1}{2}\ln u + \ln(1+u)\big|_{u=1} = -\frac{1}{2} + \ln 2 > 0$. ∎

LEMMA A.5. *Let $Y = \sum_i \alpha_i X_i$, where $\alpha_i \in \{0,1\}$ and $X_i \sim Ber(\eta)$ are independent for some $\eta \in (0, 1/50)$. Then for every $d \geq \sum_i \alpha_i$*

$$\Pr[Y \geq d/2] \leq (3\eta)^{d/5}.$$

Proof. We have $\alpha_i X_i \leq 1$ with probability 1, and

$$v = \sum_i \mathbb{E}[(\alpha_i X_i)^2] \leq \sum_i \mathbb{E}[\alpha_i X_i] = \eta d.$$

Noting that the function $vh(\frac{x}{v})$ is monotone decreasing in v for any $x \geq 0$ (by applying Claim A.3 after rescaling v), we get, letting $t = (1/2 - \eta)d$ in Bennett's inequality (Theorem A.2) with $b = 1$, $t = (1/2-\eta)d$, and v,

$$\begin{aligned}
\Pr\left[\sum_i \alpha_i X_i \geq d/2\right] &= \Pr\left[\sum_i \alpha_i X_i - \mathbb{E}[\alpha_i X_i] \geq (1/2-\eta)d\right] \\
&\leq \exp\left(-v \cdot h\left(\frac{(1/2-\eta)d}{v}\right)\right) \\
&\leq \exp\left(-\eta d \cdot h\left(\frac{(1/2-\eta)d}{\eta d}\right)\right) \\
&= \exp\left(-\eta d \cdot h\left(\frac{1/2-\eta}{\eta}\right)\right)
\end{aligned}$$

Noting that $h(u) \geq \frac{1}{2}u\ln u$ for all $u \geq 0$ by Claim A.4, we get

$$\begin{aligned}
\Pr\left[Y \geq d/2\right] &\leq \exp\left(-\frac{1}{2}\eta d \cdot \frac{1/2-\eta}{\eta} \cdot \ln\left((1/2-\eta)/\eta\right)\right) \\
&\leq \exp\left(-(d/5)\cdot\ln(1/3\eta)\right) \\
&\leq (3\eta)^{d/5}
\end{aligned}$$

∎

LEMMA A.6. *Let $E_1, \ldots, E_k, Z_1, \ldots, Z_k$ be arbitrarily correlated random variables. Let $\widetilde{\eta} \in (0, e^{-5k})$ and the positive integers $(q_i)_{i=1}^k$ be such that, for all $i \in [k]$, $E_i \in \{0,1\}$, $\mathbb{E}[E_i] \leq \widetilde{\eta}^{q_i/5}$, $Z_i \in [0,1]$, and $\mathbb{E}[Z_i] \leq \widetilde{\eta}$. Then*

$$\mathbb{E}\left[\prod_{i=1}^k (1 + Z_i + q_i E_i)\right] \leq 1 + 3^k \widetilde{\eta}^{1/5}.$$

Proof. We expand the product and show that each monomial is small in expectation. There are 3^k terms, each of the form

$$\mathbb{E}\left[\prod_{i\in S_1} Z_i \prod_{j \in S_2} q_j E_j\right]$$

for disjoint sets $S_1, S_2 \subseteq [k]$. First, $S_1 = S_2 = \emptyset$ corresponds to the leading 1 term. Next, when $S_1 \neq \emptyset$ but $S_2 = \emptyset$, we have for any $i^* \in S_1$ that

$$\mathbb{E}\left[Z_{i^*} \cdot \prod_{i \in S_1 \setminus \{i^*\}} Z_i\right] \leq \mathbb{E}[Z_{i^*}] = \widetilde{\eta}.$$

Finally, if $S_2 \neq \emptyset$, let $j^* \in S_2$ be of maximal q_{j^*}, and define $q = q_{j^*}$. Then

$$\mathbb{E}\left[\prod_{i \in S_1} Z_i \prod_{j \in S_2} q_j E_j\right] \leq \left(\prod_{j \in S_2} q_j\right) \Pr[E_{j^*}] \leq q^k \cdot \widetilde{\eta}^{q/5}$$

We now upper bound the last term on the rhs. The function

$$f(q) = q^k \cdot \widetilde{\eta}^{q/5}$$

has, for all positive integers q,

$$f(q+1)/f(q) = (1 + 1/q)^k \widetilde{\eta}^{1/5} \leq 2^k \widetilde{\eta}^{1/5} < (e^{5k}\widetilde{\eta})^{1/5} \leq 1.$$

Therefore $f(q)$ is maximized (over positive integers) when $q = 1$ and $f(q) = \widetilde{\eta}^{1/5}$. Therefore every term other than the leading 1 is at most $\widetilde{\eta}^{1/5}$. There are 3^k terms, giving the result. □

B Generating Timestamps in the Stream

In this section we show how an algorithm can generate n timestamps in a streaming manner, corresponding to drawing n uniform random variables from $(0, 1)$ and then presenting each in order with $\text{poly}(1/n)$ precision, using $O(\log n)$ bits of space.

Let $(\mathbf{X}_i)_{i=1}^n$ denote n variables drawn independently from $\mathcal{U}(0, 1)$ and then ordered so that $\mathbf{X}_i \leq \mathbf{X}_{i+1}$ for all $i \in [n-1]$. By standard results on the order statistics (see e.g. page 17 of [DN03]), the distribution of $(\mathbf{X}_i)_{i=j+1}^n$ depends only on $\mathbf{X}_j$, and in particular they are distributed as drawing $(n-j)$ samples from $(\mathbf{X}_j, 1)$.

So then, to generate $(\mathbf{X}_i)_{i=1}^n$ with $\text{poly}(1/n)$ precision in the stream it will suffice to, at each step $i+1$, use $\mathbf{X}_i$ to generate $\mathbf{X}_{i+1}$ (as sampling from the minimum of k random variables to $\text{poly}(1/n)$ precision can be done in $O(\log n)$ space). We will only need to store one previous variable at a time, to $\text{poly}(1/n)$ precision, and so this algorithm will require only $O(\log n)$ space.

C Proofs of Corollary 1.5 and Corollary 1.6

Proof of Corollary 1.5: By Theorem 1.3 the output of SamplesWithReset is $2^{-\Omega(D)}$-close in total variation distance to a samples of ε-approximate samples of the k-step walk in G. We first analyze the algorithm assuming full independence of $(v^i)_{i \in [a]}$. Under this assumption we have

$$(1 - O(\varepsilon)) \cdot \text{rp}(G) \leq \mathbb{E}_{\mathbf{v}^i}[v_k^i = v_0^i] \leq (1 + O(\varepsilon)) \cdot \text{rp}(G),$$

and get by Chebyshev's inequality, using that $a = \frac{D}{\varepsilon^2}$ for a sufficiently large constant $D > 0$,

$$\Pr\left[\left|\frac{1}{a}\sum_{i \in [a]} \mathbb{I}[v_k^i = v_0^i] - \text{rp}(G)\right| > \varepsilon\right] \leq 99/100.$$

Since the true $(v^i)_{i \in [a]}$ are $2^{-\Omega(D)}$-close to independent, the success probability is lower bounded by $9/10$ as long as D is larger than an absolute constant, as required. □

Proof of Corollary 1.6: We assume that the walks output by the invocation of ApproxPageRank(Algorithm 2) are independent, and account for the $2^{-\Omega(D)}$ additional term in the failure probability due to the total variation distance to independence at the end of the proof.

Define the truncated PageRank vector $\bar{p}_\alpha$ by

$$\bar{p}_\alpha = \sum_{k=0}^{\lceil \frac{2}{\alpha} \log 1/\varepsilon \rceil} \alpha(1-\alpha)^k M^k \cdot \frac{\mathbb{1}}{n},$$

and note that under the full independence assumption of the walks we get that the estimator $\widehat{p}$ computed by Algorithm 2 satisfies

$$\mathbb{E}[\widehat{p}] = \bar{p}_\alpha(T).$$

We thus get by Chebyshev's inequality, using that $a = \frac{D}{\varepsilon^2}$ for a sufficiently large constant $D > 0$,

$$\Pr\left[|\widehat{p} - \bar{p}_\alpha(T)| > \varepsilon/2\right] \leq 99/100.$$

We now note that

$$\begin{aligned}
|p_\alpha(T) - \bar{p}_\alpha(T)| &\leq \|p_\alpha - \bar{p}_\alpha\|_1 \\
&= \left\| \sum_{k > \frac{2}{\alpha} \log 1/\varepsilon} \alpha(1-\alpha)^k M^k \cdot \frac{\mathbb{1}}{n} \right\|_1 \\
&\leq (1-\alpha)^{\frac{2}{\alpha} \log 1/\varepsilon} \\
&\leq \varepsilon/2
\end{aligned}$$

since $\varepsilon < 1/2$ by assumption of the corollary.

It remains to note that since the walks output by the invocation of ApproxPageRankare $2^{-\Omega(D)}$-close to independent, we have by combining the above two bounds that $|\widehat{p} - p_\alpha(T)| \leq \varepsilon$ with probability at least $99/100 - 2^{-\Omega(D)} \geq 9/10$ as long as D is larger than an absolute constant, as required.

Finally, we verify that the preconditions of Theorem 1.3 are satisfied. We invoke Theorem 1.3 with $k = \left\lceil \frac{2}{\alpha} \log(1/\varepsilon) \right\rceil$. Letting c' be the constant from Theorem 1.3, we get, using the assumption that $\varepsilon = 2^{-o(\sqrt{\log n})}$, that

$$\min\left\{ \frac{c' \log n}{\log(1/\varepsilon)}, \sqrt{c' \log n} \right\} = \sqrt{c' \log n}.$$

Thus, using the assumption of the corollary that $\frac{1}{\alpha} \leq \frac{\sqrt{\log n}}{4 \log(1/\varepsilon)}$, we get

$$k = \left\lceil \frac{2}{\alpha} \log(1/\varepsilon) \right\rceil \leq \sqrt{c' \log n},$$

as required. □

D Proof of Lemma 6.3

In this section we give a proof of Lemma 6.3 through a standard information-theoretic argument. We will need the following definitions and results from information theory (see, e.g. [CT91]).

For random variables X, Y, Z, with $p_x = \Pr[X = x]$, define

Entropy $H(X) = \mathbb{E}_X[-\log p_X]$. For $p \in [0,1]$ we write $H(p)$ for the entropy of a Bernoulli random variable with parameter p.

Conditional Entropy $H(X|Y) = \mathbb{E}_Y[\mathbb{E}_X[-\log p_X|Y]]$.

Mutual Information $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$

Conditional Mutual Information $I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$

We will use the fact that $H(p)$ is concave and uniquely maximized at $p = 1/2$, the fact that $I((X_i)_{i=1}^n; Y) = \sum_{i=1}^n I(X_i; Y|(X_j)_{j=1}^i)$, and Fano's data processing inequality.

THEOREM D.1. (FANO'S INEQUALITY) *Let X, Y be discrete random variables, and $P_e = \Pr[g(Y) \neq X]$, where g is some function on the support of Y and X is supported on $\mathscr{X}$. Then*

$$H(P_e) + P_e(\log|\mathscr{X}| - 1) \geq H(X|Y)$$

LEMMA D.2. *Let $\varepsilon > 0$ be any constant. Any protocol that solves* INDEX_n *on a uniform input with probability $1/2 + \varepsilon$ requires $\Omega(n)$ communication in expectation.*

Proof. Suppose we had such a protocol. Then for each $i \in [n]$, with X as x_i, the i^{th} bit of Alice's input, and Y as the message M she sends, and P_e as the probability that Bob gives the wrong answer when $I = i$, we have

$$H(P_e) \geq H(X|Y)$$

as X_i is supported on two elements. So for each $i \in [n]$, if p_i is the probability Bob *succeeds* when $I = i$, we have

$$H(x_i|M) \leq H(1 - p_i) = H(p_i)$$

Then, as the protocol succeeds with probability $1/2+\varepsilon$, and Bob's index I is uniform on $[n]$, $1/2+\varepsilon = \sum_{i=1}^n p_i$, so

$$\begin{aligned}
I(x; M) &= \sum_{i=1}^n I(x_i; M|(x_j)_{j=1}^i) \\
&= \sum_{i=1}^n \big(H(x_i|(x_j)_{j=1}^i) - H(x_i|M, (x_j)_{j=1}^i))\big) \\
&= \sum_{i=1}^n \big(1 - H(x_i|M, (x_j)_{j=1}^i)\big) && \text{as the } x_i \text{ are independent} \\
&\geq n - \sum_{i=1}^n H(x_i|M) \\
&\geq n - \sum_{i=1}^n H(p_i) \\
&\geq n\left(1 - H\left(\sum_{i=1}^n p_i\right)\right) && \text{by concavity} \\
&= n(1 - H(1/2 + \varepsilon)) \\
&= \Omega(n)
\end{aligned}$$

as ε is a non-zero constant. So in particular $H(M) = \Omega(n)$ and so M is at least n bits on average.

□

E Lower Bound for Chosen Vertices

In this appendix, we prove that, if instead of sampling a walk from a randomly chosen vertex we need to sample a walk from a chosen vertex, any algorithm that gives better than a $1/2 - 2^{-2-\lfloor \frac{k}{2} \rfloor}$-approximation to the distribution of length $k \geq 2$ random walks on *undirected* graphs needs at least $\Omega(n)$ space.

THEOREM E.1. *For any constant $\varepsilon < 1/4 - 2^{-2-\lfloor \frac{k}{2} \rfloor}$, the following holds for all $k \geq 2$ and all n: there is a family of (undirected) graphs with n vertices and edges such that any algorithm that, when given a specified vertex v and then the graph as a random order stream, ε-approximates the distribution of length-k random walks on graphs drawn from the family uses $\Omega(n)$ bits of space, with a constant factor depending only on ε.*

The proof will be a simplified version of the hardness proof for digraphs given in Section 6. We will prove that any algorithm giving such an approximation gives a protocol solving INDEX_{n-3} on a random instance with probability a constant factor greater than 1/4. We restate the lower bound on INDEX_n here for convenience. Recall that for INDEX_n Alice's input is a string $x \in \{0,1\}^n$, Bob's an index $I \in [n]$, and their objective is for Alice to send Bob a message such that Bob can determine x_I.

LEMMA E.2. *Let $\varepsilon > 0$ be any constant. Any protocol that solves INDEX_n on a uniform input with probability $1/2 + \varepsilon$ requires $\Omega(n)$ communication in expectation.*

E.1 Graph Distribution We start by defining a distribution on length-$(n-3)$ undirected graph streams that Alice and Bob can construct (with a prefix belonging to Alice and a postfix belonging to Bob) using their inputs to INDEX_{n-3}.

We will show that the graph streams correspond to randomly choosing a graph and then uniformly permuting its edges, and any algorithm that generates a distribution that is ε-close to the distribution of walks starting at a specified vertex for some constant $\varepsilon < 1/4 - 2^{-2-\lfloor \frac{k}{2} \rfloor}$ will be able to use this to solve INDEX_{n-3}.

Vertices. There will be 1 vertex a, $n-3$ vertices $(b_i)_{i=1}^{n-3}$, and two vertices c_0, c_1. a will be in a 2-edge component with one of c_0, c_1.

Alice's edges. Let π be a uniformly random permutation of $[n]$, and let J be drawn uniformly from $\{0, \ldots, n\}$. These will be used to define the boundary between Bob's edges and Alice's edges.

Recall that Alice's input is a string $x \in \{0,1\}^n$. For each $i \in [J]$, Alice has the edge $b_{\pi(i)}c_{x_{\pi(i)}}$, with $b_{\pi(1)}c_{x_{\pi(1)}}$ first, $b_{\pi(2)}c_{x_{\pi(2)}}$ second, and so on.

Bob's edges Recall that Bob has the index I. His first edge will be ab_I. Then, for each $i \in \{J+1, \ldots n\}$, he has the edge $b_{\pi(i)}c_{y_{\pi(i)}}$, where y is a random n-bit string.

LEMMA E.3. *The graph stream described above is a uniformly random order graph stream.*

Proof. Fix any value of Bob's index I. If we were to randomly draw a string $z \in \{0,1\}^n$, randomly order edges $(b_i c_{z_i})_{i=1}^n$, and then insert ab_I randomly in this stream, this would give a uniformly random order graph stream (corresponding to drawing a graph from a fixed distribution and then randomly permuting its edges).

But this would also be identically distributed to our graph stream, as the strings x and y are both drawn uniformly at random from $\{0,1\}^n$. So we have a uniformly random graph stream. ∎

LEMMA E.4. *With probability 3/4, a at one end of a 2-edge path with c_{x_I} at the other end. Otherwise it is at one end of a 2-edge path with $c_{\overline{x_I}}$ at the other end.*

Proof. There is an edge between a and b_J, and with probability 1/2, $\pi(I) \leq J$, and so the only other edge incident to b_J is incident to c_{x_I}. Otherwise, it is incident to c_{y_I}, which is c_{x_I} with probability 1/2, as y is drawn at random. ∎

We will now prove that the random walk distribution from a can be used to determine the answer to INDEX_{n-3}.

LEMMA E.5. *Let $\mathcal{A}$ be any algorithm using S space such that, when given a sample from the distribution on graph streams above, the output of $\mathcal{A}$ is ε-close in total variation distance to sampling a k-step random walk from a, where $k \geq 2$. Then there is a protocol for INDEX_{n-3} on uniform inputs that uses S space and succeeds with probability $\frac{3}{4} - 2^{-2-\lfloor \frac{k}{2} \rfloor} - \varepsilon$.*

Proof. The protocol will be as follows:

1. Alice and Bob use their INDEX input to construct a corresponding random graph stream.

2. Using S bits of communication from Alice to Bob, they run $\mathcal{A}$ on the stream.

3. If the walk output by $\mathcal{A}$ ever reaches c_b for some $b \in \{0,1\}$, output i. Otherwise output a random $b \in \{0,1\}$.

Now, by Lemma E.4, a is at one end of a 2-edge path with one of c_0, c_1 at the other end, and there is a $3/4$ chance it is c_{x_I}. So there is a

$$1 - 2^{-\lfloor \frac{k}{2} \rfloor}$$

probability any walk from a reaches a vertex in c_0, c_1. If the walk output by $\mathcal{A}$ does this the protocol succeeds with probability $3/4$ and otherwise it succeeds with probability $1/2$. So the correct answer is returned with probability at least

$$(1 - 2^{-\lfloor \frac{k}{2} \rfloor})\frac{3}{4} + (2^{-\lfloor \frac{k}{2} \rfloor})\frac{1}{2} - \varepsilon = \frac{3}{4} - 2^{-2-\lfloor \frac{k}{2} \rfloor} - \varepsilon$$

proving the correctness of the protocol. The construction of the stream itself is done entirely with public randomness, so this gives an S bit public randomness protocol. As we are working with a fixed input distribution, this means that there is also an S bit private randomness protocol, by fixing whichever set of public random bits maximizes the probability of success over the uniform distribution. □

This then proves Theorem E.1, as any algorithm that can output a constant $\varepsilon < 1/4 - 2^{-2-\lfloor \frac{k}{2} \rfloor}$ approximation to the random walk distribution will give a protocol for INDEX_{n-3} that works with a probability at least a constant greater than $1/2$.