Pseudorandom Graphs in Data Structures

Omer Reingold[1], Ron D. Rothblum[2], and Udi Wieder[3]

[1] Microsoft Research, omer.reingold@microsoft.com
[2] Weizmann Institute, ron.rothblum@weizmann.ac.il
[3] Microsoft Research, uwieder@microsoft.com

Abstract. We prove that the hash functions required for several data structure applications could be instantiated using the hash functions of Celis *et al.* (SIAM J. Comput., 2013). These functions simultaneously enjoy short description length as well as fast evaluation time. The applications we consider are: (1) Cuckoo Hashing, (2) Cuckoo Hashing with Stash and (3) the Power of Two Choices paradigm for load balancing. Our analysis relies on a notion of sparse pseudorandom graphs that are similar to random graphs in having no large connected component and no dense subgraph. Such graphs may be of independent interest. Relating pseudorandom graphs to the two-choice paradigm relies on a very simple new proof we give (at the price of somewhat worse parameters).

1 Introduction

A common assumption in the design of data structures is the availability of perfectly random hash functions, functions that are assumed to require no space to store and take unit time to compute. A large body of work is dedicated to the removal of this assumption by designing constructive and explicit families of functions with well defined properties. In this paper we show that the family of functions put forth by Celis *et al.* [CRSW13] can be used for variants of Cuckoo Hashing, and could also be used to place balls into bins in the "power of two choices" scheme. A function from this class can be represented using $O(\log n \log\log n)$ bits and evaluated in $O((\log\log n)^2)$ operations in a unit-RAM model. In both cases this is the first construction where the description length is lower than $O(\log^2 n)$ which is the bound obtained by $\log n$-wise independent functions. The analysis is done by showing that a random graph built using these functions shares structural properties with truly random graphs, a result that could have more applications and be of independent interest.

Cuckoo Hashing: In the dictionary problem the goal is to build a data structure that represents a set S of size n taken from a universe $\mathcal{U}$. The data structure should support look-ups, as well as updates. In an influential work, Pagh and Rudler [PR04] constructed a dictionary, referred to as *cuckoo hashing* which has worst case constant lookup time, amortized expected constant update time, and uses slightly more than $2n$ words for storing n elements. Additional attractive features of cuckoo hashing are that no dynamic memory allocation is performed (except for when the tables have to be resized), and the lookup procedure queries only two memory entries which are independent and can be queried in parallel.

Cuckoo hashing uses two tables T_1 and T_2, each consisting of $r \geq (1+\epsilon)n$ words for some small constant $\epsilon > 0$, and two hash functions $h, g : \mathcal{U} \to \{0, \dots, r-1\}$. An element $x \in \mathcal{U}$ is stored either in entry $h(x)$ of table T_1 or in entry $g(x)$ of table T_2, but never in both. The lookup procedure is straightforward: when given an element $x \in \mathcal{U}$, query the two possible memory entries in which x may be stored. For insertions, Pagh and Rudler [PR04] suggested the "cuckoo approach", kicking other elements out until every element has its own "nest". Insertion is shown to run in expected constant time and w.h.p no more than $O(\log n)$ time when the functions h and g are truly random, that is, they sample an element in $[r]$ uniformly and independently.

Random graphs: A random graph $G(m, n)$ over m nodes is obtained by sampling uniformly and independently n unordered pairs of nodes.[4] The sampling is done without replacement so we allow multiple edges and self loops. When the graph is sparse, i.e. when $n < (1-\epsilon)m/2$ for some $\epsilon > 0$, it is known that $G(m, n)$ is likely to have some distinctive properties: connected components are small and each connected component is sparse (we leave for now the exact definition of sparseness). These properties had been used in the analysis of Cuckoo Hashing in a natural way.

[4] We use a non-standard notation where m denotes the number of nodes and n the number of edges for sake of consistency with the applications to data structures described below.

To analyze cuckoo hashing, let G be a graph with $(1+\epsilon)2n$ nodes, associated with the locations of the two tables T_1, T_2. For each item x in the set S that is represented by the data structure, add an edge $(h(x), g(x))$ (the graph as described here is bipartite, but this makes little difference in the analysis). Now it is not hard to observe that insertion succeeds as long as each connected component in G has at most one cycle. Pagh and Rudler proved that this is indeed the case w.h.p in this parameter regime. Furthermore, the expected and worst-case sizes of a connected component determine the expected and worst-case insertion time.

It follows that one way to prove that a specific (explicit) family of hash functions $\mathcal{H}$ could be used for cuckoo hashing is to show that the graph obtained by sampling h, g and constructing G as above, has the same structural properties as a random graph. The goal is that the family $\mathcal{H}$ be as small as possible (so that the description length of each function is short) and that the functions be easy to evaluate. In this paper we show how to instantiate this approach with the function of [CRSW13]. We also find a surprising application for it to the power of two-choice paradigm.

The power of two choices: In the Greedy[d] process (sometimes called the d-choice process), balls are placed sequentially into d bins with the following rule: each ball is placed by uniformly and independently sampling d bins and assigning the ball to the least loaded of the d bins. We are interested in the load, measured in number of balls, of the most loaded bin. The case $d = 1$, when balls are placed uniformly at random in the bins, is well understood. When n balls are thrown, the bin with the largest number of balls will have $\Theta(\log n/\log\log n)$ balls w.h.p. In an influential paper Azar *et al.* [ABKU99] showed that when $d > 1$ and n balls are thrown the maximum load is $\log\log n/\log d + O(1)$ w.h.p. The proof in [ABKU99] uses an inductive argument which relies on the assumption of full randomness.

1.1 Previous Work

A natural explicit family of functions that is useful in many applications is that of $\log n$-wise (almost) independent functions. These are functions where every $\log n$ evaluations are (almost) uniformly distributed. A $\log n$-wise independent family could be represented using $O(\log^2 n)$ bits, say by specifying the coefficients of a degree $\log n$ polynomial, and naively could be evaluated in $O(\log n)$ time. Recently, Meka *et al.* [MRRR13] constructed a family of $\log(n)$-wise almost independent *Boolean* hash function that can be evaluated in $O(\log\log n)$ time.

Intuitively it is clear why $O(\log n)$-independence should suffice for Cuckoo Hashing; the connected components of $G(m, n)$ are of logarithmic size, so in some sense all the ‘interesting’ events occur over a logarithmic number of variables. Indeed, the proof in [PR04] carefully counts subgraphs of a certain type (long paths and large trees) and eliminates them one by one using a union bound. Such an approach could be transformed almost as-is to handle $\log n$-wise independent functions.

A similar result for the 'power of two' case is not so straightforward, the proof in [ABKU99] uses an inductive argument that relies on the assumption of full randomness. A different proof was discovered by Vöcking' [V03] which uses a construct called 'witness trees'. These are essentially labeled subgraphs of a naturally defined random graph. Vöcking's goal was to analyze a variant of the process called GoLeft, but as a byproduct he implicitly showed that $O(\log n)$-wise independent functions could be used. We show a different and much simpler way of using random graphs to prove this result - albeit with a larger constant in front of the $\log \log n$ term.

It is natural to ask whether one can construct a family of hash functions that improves both the time to evaluate a function and the space complexity (i.e., the description length). Ideally we would like functions that can be represented by $O(\log n)$ bits and evaluated in $O(1)$ time in the unit RAM model. It is worth mentioning that there exist important data structures for which such functions are known. A notable example is that of the linear probing data structure where Pagh et.al. [PPR11] show that any 5-wise independent hash function suffices. Mitzenmacher and Vadhan [MV08] showed that pairwise independent functions could be used for Cuckoo Hashing, providing the keys themselves come with sufficient entropy. For arbitrary input it is not known whether there exists a constant k such that any k-wise independent construct could be used for Cuckoo Hashing or balls-into-bins, and this remains an interesting open problem.

There is an impressive body of literature presenting functions with a running time of $O(1)$, at the expense of the space complexity. Starting with Siegel [Sie04], and continuing with e.g. Dietzfilbinger and Woelfel [DW03], Patrascu and Thorup [PT12] and Thorup [Tho13]. All these solution need $n^{\Theta(1)}$ bits of space to represent a function. In this paper we are more concerned with the space complexity. Can we construct a family of functions where a function can be represented using $o(\log^2 n)$ bits? A step in this direction is the load-balancing functions of [CRSW13] which are the starting point of our work.

1.2 Our contributions

Our main contribution is to show that the family of functions put forth in [CRSW13] (henceforth, the CRSW hash functions) can be used for cuckoo hashing and for the power of two choices. The space needed to represent a function is $O(\log n \log \log n)$ bits, so these functions constitute the only family we are aware of that suffices for these applications with space complexity lower than that of $\log n$-wise independence. Additionally, it was recently shown by Meka *et al.* [MRRR13] that hash function from this family can be evaluated efficiently, in $O((\log \log n)^2)$ time.

Our analysis is done by proving that the random graph derived by this family has structural properties similar to those of a truly random graph; a result that may find further applications.

Let $\mathcal{U}$ be a universe of size $\mathsf{poly}(n)$ and let $\mathcal{H}$ be a family of functions from $\mathcal{U} \to V$. Every set $S \subset \mathcal{U}$ and pair of functions $h, g \in \mathcal{H}$ that are sampled uniformly and independently from $\mathcal{H}$, define a graph whose node set is V and with the edge set $\{(h(x), g(x)) : x \in S\}$.

Our main result is that if $\mathcal{H}$ is the CRSW hash function family, $|S| = n$ and $|V| = O(n)$, then the graph G has the following two properties:

– **Small Components:** For every $v \in V$, let $C(v)$ be the connected component of v in G. Then the expected size of $C(v)$ is $O(1)$, and is $O(\log n)$ w.h.p.
– **Sparse Components:** W.h.p the number of edges in $C(v)$ is bounded by $2|C(v)|$ (where 2 is an arbitrary constant that can be made as close to 1 as we like).

As alluded to before, these two properties underlie the analysis of cuckoo hashing and could be used to prove double logarithmic bounds on the load of Greedy$[d]$. We show that $\mathcal{H}$ can be used for the following data structures.

1. **Cuckoo Hashing:** A cuckoo hashing scheme with $O(n)$ memory cells and two items per bucket. The idea to allow more than one item in a memory cell is due to Diezfelbinger and Weidling [DW07]. They show that this approach increases the memory utility of the scheme. We note that we do not match the space efficiency obtained with fully random functions and observe that it is not known whether $\log n$-wise independent function obtain it either.
2. **Cuckoo Hashing with Stash:** One drawback of cuckoo hashing is that with some fixed polynomially small probability an insertion operation might fail. Kirsch *et al.* showed in [KMW09] that allocating a special location in memory (stash) which may hold any data item and is searched on every look-up, indeed reduces the failure probability significantly, even when the stash is of constant size. Aumüller *et al.* [ADW12] show that $O(\log n)$-wise independent functions would work in this case. Our functions also benefit from the use of stash, that is, using only a constant size stash, we can guarantee an arbitrarily small polynomial probability of failure.
3. **The Power of Two Choices:** when n balls are placed in n bins using two random functions from $\mathcal{H}$, the maximum load is $O(\log\log n)$ w.h.p. We observe that we lose a constant factor over truly random functions in which case the load is $\log\log n + O(1)$. On the plus side our proof is remarkably simple (and of course also holds for the truly random case). Our simple idea is as follows: assume that instead of throwing n balls to n bins we throw $n/10$ balls. Now consider the graph with n vertices (corresponding to the bins) and $n/10$ edges (where an edge corresponding to a ball is adjacent to the two bins it can land at). The observation is that this graph is now sparse enough (or in other words, is in the sub-critical regime) and can have the pseudorandom properties of no large connected component and no dense subgraph. A very simple argument, based on the 'witness trees' of Vöcking' [V03] implies an $O(\log\log n)$ load. We finally give a simple reduction from the case of n balls to n bins to the case of $n/10$ balls to n bins. The reduction also increases the maximum load by a constant factor (in fact this is where the main loss in parameters comes from). It is interesting to note that while the parameters are a bit weaker the final result is a bit stronger in the additional power it gives the adversary: the result holds even if (after the hash functions are made public) the adversary has full control over the order in which the balls arrive in each batch of $n/10$ balls.

1.3 Main Technical Ideas for the Pseudorandom Graphs

We sketch the main idea behind the CRSW functions. Each function maps an item from $\mathcal{U}$ to $[n]$, so the output is $\log n$ bits long, but it is often convenient to think of the output as one of n bins. The function h is obtained by sampling $d = \Theta(\log \log n)$ *different* functions and concatenating their output so that $h(x) = h_1(x) \circ h_2(x) \circ \cdots \circ h_d(x)$, where $\circ$ denotes concatenation. Each h_i is a k_i-independent function with an output length of ℓ_i bits, such that $\sum \ell_i = \log n$. It was shown in [CRSW13] that parameters could be chosen so that the total memory needed to represent h is $\log n \log \log n$ bits. The main idea is that as i increases the independence also increases (exponentially), so $k_1 = O(1)$ while $k_d = O(\log n)$. On the other hand, the output ℓ_i decreases with i, so that the space needed to represent the function remains small.

As shown in [CRSW13], the key property that these functions enjoy is that they are 'balancing'. Let $S \subset \mathcal{U}$ be a set of size n. Fix some $j \leq d$ and consider the first j functions $h_1, ..., h_j$. Their output is concatenated to produce a string of $\sum_{i \leq j} \ell_i$ bits. The guarantee is that w.h.p h distributes the n values of S across these $2^{\sum_{i \leq j} \ell_i}$ values as well as a random function. In particular, when $j = d$ we get that the most heavily loaded of the n bins receives at most $O(\log n / \log \log n)$ items (which was the main result of [CRSW13]).

How can we use this property to argue on the structure of graphs? Our goal is to show that certain types of subgraphs are not likely to occur. Specifically, we want to exclude logarithmically large trees and dense subgraphs. As each of these graphs have at most $O(\log n)$ edges, we could select h and g to be $O(\log n)$-wise independent and rule out the existence of each specific graph. If the parameters are set correctly, a union bound over all possible "bad" graphs would give the desired result. However, when we sample h and g from the CRSW family of functions then we only have $\log n$ wise independence for a short suffix of these functions output, which is insufficient for this analysis.

Instead, we apply the following strategy. Consider h as in the CRSW functions. It is the concatenation of $h_1 \dots h_d$. Assume we want to rule out bad graphs with t edges (where t is $O(\log n)$ but can also be as small as a constant). Let $j < d$ be minimal such that $h_{j+1} \dots h_d$ are all at least t-wise independent. Let $\ell' = \sum_{i \leq j} \ell_i$, so ℓ' is the length of the concatenated output of the first j functions. We can take the vertices of G and collapse every $2^{j'}$ vertices which share the same prefix of length j' into one 'super vertex'. Now, the load balancing property tells us that the degree of each of these super vertices should be close to its expectation. We show that such load balancing is all we need from $h_1, \dots, h_j$. Now $h_{j+1} \dots h_d$ are not only independent of $h_1, \dots, h_j$ but also have sufficient independence for us to complete the proof.

Organization. In Section 2 we recall some standard notation and facts (an overview of the CRSW hash function is provided in Appendix A). In Section 3 we show how the CRSW hash functions can be used to produce graphs with strong random-like properties. In Section 4 we describe our applications to cuckoo hashing. Finally, in Section 5 we describe our applications to the power of two choice.

2 Preliminaries

We start by introducing some standard notations and definitions.

- All graphs are multigraphs. That is, we allow self loops and multiple edges.
- For a set S we denote by $x \in_R S$ a random variable x that is uniformly distributed in S.
- For a string $x \in \{0,1\}^n$ and an integer $\ell \in \{0,\dots,n\}$, we denote by $pref_\ell(x)$ the ℓ-bit prefix of x.

Loosely speaking, a k-wise (almost) independent hash function, is a function that looks (almost) random on any k coordinates. Formally, let $m \leq u$ be integers such that m is a power of two.

Definition 2.1. *A collection of functions $\mathcal{H} = \{h : [u] \to [m]\}$ is k-wise δ-dependent if for every distinct $x_1,\dots,x_k \in [u]$ the distribution $(h(x_1),\dots,h(x_k))$, where $h \in_R \mathcal{H}$, is δ-close to the uniform distribution over $[m]^k$.*

We will use a result of Naor and Naor [NN93], showing that there exist k-wise δ-dependent hash functions with a short description length:

Theorem 2.2 ([NN93]). *There exists a family of k-wise δ-dependent functions $\mathcal{H} = \{h : [u] \to [m]\}$ where each function can be described using $O(\log u + k \log m + log(1/\delta))$ bits.*

The following tail inequality is useful when analyzing k-wise independent hash functions:

Lemma 2.3 (Tail inequality for k-wise independence. See, e.g., [CRSW13, Lemma 2.2]). *Let $X_1,\dots,X_n \in \{0,1\}$ be $2k$-wise δ-dependent random variables, for some $k \in \mathbb{N}$ and $\delta \in [0,1)$, and let $X = \sum_{i=1}^n X_i$ and $\mu = \mathbf{E}[X]$. Then, for any $t > 0$ it holds that $\Pr[|X-\mu| > t] \leq 2\left(\frac{2nk}{t^2}\right)^k + \delta\left(\frac{n}{t}\right)^{2k}$.*

We end this section by recalling the well-known fact that the number of (unlabeled) trees of size n grows exponentially with n:

Lemma 2.4 (See, e.g., [LPV03, Theorem 8.5.1]). *The number of unlabeled trees of size n is at most 4^{n-1}.*

3 Pseudorandom Graphs

Let $n \leq m \leq u$ be integers as in Appendix A (in particular $m = \alpha n$ for some sufficiently large constant $\alpha > 1$, and m is a power of two). We consider graphs that are constructed by taking two hash functions and adding edges that correspond to evaluations of the hash functions in the following natural way:

Definition 3.1. *Let $h, g : [u] \to \{0,\dots,m-1\}$ be functions and let $S \subseteq [u]$ be a set of size n. We define the (multi-)graph $G_{h,g}(S) \stackrel{\text{def}}{=} \big(\{0,\dots,m-1\}, E\big)$, where $E \stackrel{\text{def}}{=} \{(h(x), g(x)) : x \in S\}$.*

Let $\mathcal{H}_{\mathsf{CRSW}}$ be the CRSW hash function family (see Appendix A and in particular Construction A.1 for a full description). In this section we show that for every $S \subseteq [u]$, with high probability, a random graph $G_{h,g}(S)$, where h and g are chosen at random from $\mathcal{H}_{\mathsf{CRSW}}$, does not contain any large connected subgraphs nor small connected subgraphs that are "dense". This is formalized by Theorems 3.2 and 3.3 below. In addition, we prove in Theorem 3.4 that in *expectation* each connected component has constant size.

Theorem 3.2. *For every constant $c_1 > 0$ and for any sufficiently large constant $c_3 > 0$, there exists a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every $S \subseteq [u]$, with probability $1 - n^{-c_1}$ over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$, the graph $G_{h,g}(S)$ does not contain a connected subgraph of size greater than $c_3 \log n$.*

Theorem 3.3. *For every constant $c_2 > 0$ there exists a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every $S \subseteq [u]$, with probability $1 - O(1/n)$ over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$, for every connected subgraph (V, E) of the graph $G_{h,g}(S)$ it holds that $|E| < (1 + c_2) \cdot |V|$.*

Theorem 3.4. *There exists a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every $S \subseteq [u]$ and for every $x \in S$, the expected size of the connected component that contains x (in $G_{h,g}(S)$) is $O(1)$.*

Remark. In the theorem statements, by a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$, we mean that we can set the constants in Construction A.1 so that the k_i's and d are sufficiently large and so that the δ_i's are sufficiently small.

Before the actual proof (which, due to lack of space, is deferred to Appendix B), we first give a high level overview of some of the key ideas. To prove Theorems 3.2 to 3.4 we will consider appropriate connected graphs (that are either too large or too dense) and bound the probability that they appear in $G_{h,g}(S)$. The two theorems follow by taking a union bound over all such graphs.

In order to bound the probability that a particular connected graph (V, E) appears in $G_{h,g}(S)$ we consider blocks of roughly $m^{1/|E|}$ consecutive vertices in $G_{h,g}(S)$. Observe that vertices in each such block all have the same $(1 - 1/|E|) \cdot \log(m)$-bit prefix and therefore, by the load balancing guarantee of $\mathcal{H}_{\mathsf{CRSW}}$ (for the appropriate prefix length), the number of edges that are connected to each block is roughly $n^{1/|E|}$ (i.e., as it should be if the functions were truly random). Moreover, the internal edges inside each block depend only on the $\log(m)/|E|$-bit suffix of the functions, which are $|E|$-wise (almost) independent. Since we consider events that involve at most $|E|$ items, we can treat this internal mapping inside the blocks as though it were fully random. These two properties allow us to bound the probability that connected graphs (V, E) that are either too big (i.e., $|V| = \Omega(\log n)$) or too dense (e.g., $|E| > 2|V|$).

Due to lack of space the proofs of Theorems 3.2 to 3.4 are deferred to Appendix B.

4 Cuckoo Hashing

Let $n \leq m \leq u$ and $\alpha > 1$ be as in Section 3. We consider a variant of cuckoo hashing where (as usual) we want to store n items in $m = \alpha n$ bins, but each bin can contain *two* items (in contrast to "vanilla" cuckoo hashing in which each bin holds at most one ball). As in vanilla cuckoo hashing, two hash functions $h, g : [u] \to [m]$ are chosen at random and each item x is stored either in bin $h(x)$ or bin $g(x)$. The hash functions h and g are chosen at random from the collection $\mathcal{H}_{\mathsf{CRSW}}$ (see Appendix A).

At any given point, if the data structure contains the items $S \subseteq [u]$, we can consider the cuckoo graph $G_{h,g}(S)$. Recall that this graph contains an edge $(h(x), g(x))$ for every $x \in S$. We view the graph as being directed where edges are directed from $h(x)$ to $g(x)$ if x is contained in the bin $h(x)$.

To insert an element x, we start a depth first search (DFS) starting at $h(x)$ on the (directed) cuckoo graph. When taking a step on edge $(h(y), g(y))$ we switch the orientation of the edge, which corresponds to moving the item, say, from $h(y)$ to $g(y)$. Once we reach a vertex that has out-degree less than two, we stop the process (since the current item can be stored in the bin that corresponds to this vertex while maintaining a load of at most two items).

Theorem 4.1. *There exists a setting of parameters for $\mathcal{H}_{\mathsf{CRSW}}$ (see Construction A.1) such that for every subset $S \subseteq [u]$ of size n, if we choose $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$ then, the items in S can be allocated to m bins such that:*

1. *Every item $x \in S$ is contained either in the bin $h(x)$ or $g(x)$.*
2. *Each bin holds at most two items.*
3. *With probability $1 - O(1/n)$ insertion takes time $O(\log n)$, and the expected insertion time is $O(1)$.*

Proof. Items 1 and 2 above follow directly from the construction. We proceed to prove that item 3 holds.

Let c_3 be a sufficiently large constant. We say that the graph $G_{h,g}(S)$ is good if it contains no connected subgraph (V, E) such that $|V| \geq c_3 \log n$ or $|E| \geq \frac{3}{2}|V|$. By Theorem 3.2 and Theorem 3.3, the graph $G_{h,g}(S)$ is good, with probability $1 - O(1/n)$ (over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$).

Claim 4.2. *Let $S \subseteq [u]$ be a set of n items and suppose that the graph $G_{h,g}(S)$ is good. Then there is a vertex v in the connected component of $h(x)$ in the directed cuckoo graph with out-degree less than 2.*

Proof. Suppose that some of the items have been allocated and let $x \in S$ be an item that has not yet been allocated. Let $G_u = G_{h,g}(S)$ be the *undirected* cuckoo graph and let G_d be the *directed* cuckoo graph, where edges are directed from the bin that holds an item to the bin that does not (if an item has not yet been allocated then the corresponding edge does not appear in G_d). Let V be the connected component in G_u to which $h(x)$ belongs and suppose that each vertex in V has degree 3 (in G_u). Hence, the total number of edges in V is at least $\frac{3}{2}|V|$, in contradiction to $G_{h,g}(S)$ being good.

Hence, there exists a vertex v in the connected component of $h(x)$ (in G_u) with degree at most 2. If v has out-degree less than 2 in G_d then we are done. Otherwise (i.e., if v has out-degree exactly 2 in G_d), we remove v and recursively apply the argument to the induced subgraph. □

Hence, with probability $1 - O(1/n)$ the connected component of $h(x)$ in the directed cuckoo graph has a vertex with degree less than 2 (and the graph is good) and so the insertion time is at most the time for the DFS to go over the entire connected component. Since each connected component has size at most $O(\log n)$ with $O(\log n)$ edges, we obtain the worst case guarantee.

We proceed to prove that insertion takes expected time $O(1)$. Let I_x be the insertion time of x and let C_x be the size of the connected component that contains x (both I_x and C_x are random variables).

$$\begin{aligned}\mathbf{E}[I_x] &= \Pr[G_{h,g}(S) \text{ good}] \cdot \mathbf{E}[I_x | G_{h,g}(S) \text{ good}] + \Pr[G_{h,g}(S) \text{ not good}] \cdot \mathbf{E}[I_x | G_{h,g}(S) \text{ not good}] \\ &\leq \Pr[G_{h,g}(S) \text{ good}] \cdot \mathbf{E}[I_x | G_{h,g}(S) \text{ good}] + O(1/n) \cdot O(n) \\ &= \Pr[G_{h,g}(S) \text{ good}] \cdot O(\mathbf{E}[C_x | G_{h,g}(S) \text{ good}]) + O(1) \\ &\leq O(\mathbf{E}[C_x]) + O(1) \\ &= O(1)\end{aligned}$$

where the first inequality follows from the fact that connected component of x has (in the very worst case) $O(n)$ vertices and edges and the last inequality follows from Theorem 3.4. □

4.1 Cuckoo Hashing with Constant Size Stash

Theorem 4.1 shows that the probability for a "failure" (i.e., insertion that takes super logarithmic time) when using our variant of cuckoo hashing is at most $O(1/n^2)$. For some applications it is useful to obtain a probability of error that is inversely proportional to an *arbitrarily* large polynomial.

The same problem affects standard cuckoo hashing (with one item per bin and totally random functions). Kirsch *et al.* [KMW09] suggested to solve this problem by introducing a constant size stash in which items may also be stored. The idea of [KMW09] is that if insertion takes more than logarithmic time, then the item is stored in the stash (for further details, see [KMW09]). To check whether an item $x \in [u]$ is the data structure, one simply checks if x is contained in $h(x)$, $g(x)$, or in the (constant size) stash.

In this section we show that one can decrease the error probability to be inversely proportional to an arbitrary polynomial using a constant size stash also in our variant of cuckoo hashing (i.e., with two items per bin and using the [CRSW13] hash functions).

Theorem 4.3. *For every $c_1 > 0$, there exists a constant $s > 0$ and a setting of parameters for $\mathcal{H}_{\mathsf{CRSW}}$ (see Construction A.1) such that for every subset $S \subseteq [u]$ of size n, if we choose $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$ then, the items in S can be allocated to m bins and a stash of size s such that:*

1. *Every item $x \in S$ is contained either in the bin $h(x)$ or $g(x)$, or in the stash.*
2. *Each bin holds at most two items.*
3. *With probability $1 - n^{-c_1}$ insertion takes time $O(\log n)$, and the expected insertion time is $O(1)$.*

Proof. Items (1) and (2) clearly hold and the proof that the expected insertion time is constant is similar to the proof of Theorem 4.1 (while noting that inserting into the stash takes constant time).

Let t and s be sufficiently large constants and set the parameters of $\mathcal{H}_{\mathsf{CRSW}}$ so that both Theorem B.5, Theorem B.6 and Theorem B.7 hold (with respect to $c_1' = 3c_1$ and $c_2 = \frac{1}{2}$). Hence, by the union bound, with probability n^{-c_1} over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$ it holds that (simultaneously):

(E1): Every connected subgraph (V, E) of the graph $G_{h,g}(S)$ of size $|V| \geq t$ has at most $|E| < \frac{3}{2} \cdot |V|$ edges.
(E2): Every connected subgraph (V, E) of the graph $G_{h,g}(S)$ has less than $|E| < s \cdot |V|$ edges.
(E3): Every collection of edge disjoint connected subgraphs $\{(E_1, V_1), \ldots, (E_k, V_k)\}$ of $G_{h,g}(S)$ that have $|E_i| > \frac{3}{2}|V_i|$ edges, must have $\sum_{i \in [k]} |V_i| \leq t$ total number of vertices.

In the following we condition on the events **(E1)**-**(E3)**.

If $h(x)$ is contained in a connected component (V, E) such that $|E| \leq \frac{3}{2} \cdot |V|$ then a similar analysis as in Theorem 4.1 holds. Hence, in such case x will not be stored in the stash. Note that if $|V| \geq t$, then , by **(E1)**, it holds that $|E| \leq \frac{3}{2} \cdot |V|$ and x will not be stored in the stash.

Let $\{(V_1, E_1), \ldots, (V_k, E_k)\}$ be the set of all connected components that that have density $|E_i| > \frac{3}{2} \cdot |V_i|$. Then, by **(E3)** it must be that total number of vertices $\sum_{i \in [k]} |V_i|$ is at most t. In addition, by **(E2)**, for every $i \in [k]$ it holds that $|E_i| < s|V_i|$. Hence $\sum_{i \in [k]} |E_i| \leq s \cdot t$ and all the items that are supposed to be located in these connected components can be stored in a stash of size $s \cdot t$. □

5 Two Choice Hashing

Let $n \leq m \leq u$ and α be as in Section 3. In this section we consider a data structure that holds n items in $m = \alpha n$ bins, where each bin may hold multiple items. Later, in Section 5.1 we consider the classical setting where we wish to store m items (rather than just $n < m$) in the m bins.

To store the item, two hash functions $h, g : [u] \to [m]$ are chosen at random from the collection $\mathcal{H}_{\mathsf{CRSW}}$ (see Appendix A) and each item x is stored either in bin $h(x)$ or bin $g(x)$.

When inserting, we simply insert x into the least loaded of the two bins $h(x)$ and $g(x)$ (while breaking ties arbitrarily).

As discussed in the introduction, we rely on the witness trees approach of [VÖ3], though we manage to significantly simplify the proof (at the price of somewhat worse parameters).

Lemma 5.1. *For every constant $\epsilon > 0$, there exists a setting of parameters for $\mathcal{H}_{\mathsf{CRSW}}$ such that, with probability $1 - O(1/n)$, each bin contains less than $(1+\epsilon)\log\log n$ items.*

Proof. Suppose that we are storing a set S of n items. Let G_d be the directed graph on m vertices, where each vertex corresponds to a bin and each edge corresponds to an item in the following way: for every item $x \in S$ we have an edge directed from $h(x)$ to $g(x)$ if the item is stored in bin $h(x)$. Note that the undirected version of G_d corresponds to the graph $G_{h,g}(S)$ (see Definition 3.1). We will show that if x generates a load of $\Omega(\log\log n)$, then the connected component (V, E) that $h(x)$ belongs to in G_d (and therefore also in $G_{h,g}(S)$) must be either large or dense. Since n is sufficiently smaller than m, these events were shown in Section 3 to occur with only small probability.

Fix $c_1 = 2$, let $c_2 > 0$ be a sufficiently small constant (that depends on ϵ) and let $c_3 > 0$ be sufficiently large so that Theorem 3.2 and Theorem 3.3 hold with respect to c_1, c_2 and c_3. Let $\ell \overset{\text{def}}{=} (1+\epsilon)\log\log n$. We consider the event that there exists a bin $i \in [m]$ with ℓ items and bound the probability for this event. Consider the last item x that was inserted into bin i and suppose without loss of generality that $h(x) = i$ (the case that $g(x) = i$ is completely symmetric). We iteratively mark vertices and edges in G_d by the following process.

Initially we mark only the vertex $h(x)$ and none of the edges. Next we mark the vertex $g(x)$ and the edge $(h(x), g(x))$. Observe that the bin $g(x)$ contains at least $\ell - 1$ (unmarked) items (since otherwise x would have been inserted into $g(x)$), and therefore the vertex $g(x)$ is connected to at least $\ell - 1$ unmarked edges. We mark the vertex $g(x)$. Note that each marked vertex is now connected to at least $\ell - 1$ unmarked edges. We proceed iteratively, where at iteration i each marked vertex (i.e., bin) has $\ell - i$ unmarked neighbors (i.e., items) and in the next iteration we mark all edges that correspond to $(\ell - i)$-th ball that was inserted into each of the marked bins. Let V_i be the marked vertices in iteration i and let E_i be the marked edges in iteration i. Then,

$$|V_0| = 1$$
$$|E_0| = 0$$
$$|E_{i+1}| = |E_i| + |V_i|$$

for every $i \in [\ell]$ (where the last equality follows from the fact that each edge is marked at most once and at iteration i we mark V_i new edges).

We now show that if each of the graphs (V_i, E_i) is not dense (an event that by Theorem 3.3 happens with high probability) then the last graph (V_ℓ, E_ℓ) has size exponential in ℓ. Since $\ell = \Omega(\log\log n)$, we obtain a connected component in $G_{h,g}(S)$ with $\Omega(\log n)$ size (an event that by Theorem 3.2 only happens with small probability). We proceed to the actual proof.

By Theorems 3.2 and 3.3, with probability $1-O(1/n)$ all connected subgraphs of $G_{h,g}(S)$ are not dense nor large. In particular, $|V_\ell| \le c_3 \log n$ and for every $i \in \{0,\dots,\ell\}$ it holds that $|E_i| \le (1+c_2)|V_i|$. Hence, $|E_\ell| \ge |E_{\ell-1}| + |V_{\ell-1}| \ge \frac{2+c_2}{1+c_2} \cdot |E_{\ell-1}| \ge \cdots \ge \left(\frac{2+c_2}{1+c_2}\right)^{\ell-1}$ and so the size of the last subgraph is:

$$|V_\ell| \ge \frac{1}{1+c_2} \cdot |E_\ell| \ge \frac{1}{1+c_2} \cdot \left(\frac{2+c_2}{1+c_2}\right)^{\ell-1} > c_3 \log n$$

where the last inequality follows by setting c_2 to be sufficiently small. Since such a connected subgraph of size $c_3 \log n$ only appears with probability $O(1/n)$, we conclude that the probability that there exists a bin with more than ℓ items is at most $O(1/n)$. ☐

5.1 m Items in m Bins

We first consider a slight generalization of Lemma 5.1 (that follows directly from Lemma 5.1).

Corollary 5.2. *Suppose that initially each of the m bins contains at most k items and then we add n items following the two choice paradigm, with respect to functions $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$. Then, with probability $1-O(1/n)$, each bin contains less than $k + O(\log\log n)$ items.*

Proof. By induction. Suppose that after the t-th insertion the load of each bin i is at most $k + b_i$, where b_i denotes the load of bin i after the t-th insertion in the *original* process (i.e., when all bins are initially empty). Now insert the $(t+1)$-th item, which is hashed to bins i and j. Assume that originally (i.e., when bins are initially empty) the item is stored in bin i. Hence, $b_i \le b_j$. If the item is still stored in i when the bins are first loaded with (at most) k items each then the induction trivially holds. On the other hand, if the item is stored in bin j then the load of j is smaller than $b_i + k \le b_j + k$ and therefore even after this insertion the load is at most $k + b_j$ (i.e., only k more than in the original process). ☐

By partitioning the m items into α sets of m/α balls each and applying Corollary 5.2 on each set iteratively (thereby increasing the load by an additive $O(\log\log n)$ factor on each iteration), we obtain the following theorem:

Theorem 5.3. *Suppose that we hash a set $S \subseteq [u]$ of m items into m bins following the two choice paradigm with respect to $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$. Then, with probability $1-O(1/n)$, each bin contains less than $O(\log\log n)$ items.*

Acknowledgments

We thank an anonymous reviewer for pointing out an error in a previous version of Lemma B.3.

References

ABKU99. Yossi Azar, Andrei Broder, Anna Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Computing*, 29(1):180–200, 1999.

ADW12. Martin Aumüller, Martin Dietzfelbinger, and Philipp Woelfel. Explicit and efficient hash families suffice for cuckoo hashing with a stash. In *ESA*, pages 108–120, 2012.

CRSW13. L. Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. *SIAM J. Comput.*, 42(3):1030–1050, 2013.

DW03. Martin Dietzfelbinger and Philipp Woelfel. Almost random graphs with simple hash functions. In *STOC*, pages 629–638, 2003.

DW07. Martin Dietzfelbinger and Christoph Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theor. Comput. Sci.*, 380(1-2):47–68, 2007.

KMW09. Adam Kirsch, Michael Mitzenmacher, and Udi Wieder. More robust hashing: Cuckoo hashing with a stash. *Siam J. Computing*, 39(4):1543–1561, 2009.

LPV03. László Lovász, József Pelikán, and Katalin Vesztergombi. *Discrete Mathematics — Elementary and Beyond*. Undergraduate Texts in Mathematics. Springer-Verlag, Berlin-Heidelberg-New York-Hong Kong-London-Milan-Paris-Tokyo, 2003.

MRRR13. Raghu Meka, Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Fast pseudorandomness for independence and load balancing. Manuscript, 2013.

MV08. Michael Mitzenmacher and Salil Vadhan. Why simple hash functions work: Exploiting the entropy in a data stream. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 746–755, 2008.

NN93. Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

PPR11. Anna Pagh, Rasmus Pagh, and Milan Ruzic. Linear probing with 5-wise independence. *SIAM Review*, 53(3):547–558, 2011.

PR04. Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

PT12. Mihai Patrascu and Mikkel Thorup. The power of simple tabulation hashing. *J. ACM*, 59(3):14, 2012.

Sie04. Alan Siegel. On universal classes of extremely random constant-time hash functions. *SIAM J. Comput.*, 33(3):505–543, 2004.

Tho13. Mikkel Thorup. Simple tabulation, fast expanders, double tabulation, and high independence. In *FOCS*, pages 90–99, 2013.

Vö3. Berthold Vöcking. How asymmetry helps load balancing. *J. ACM*, 50(4):568–589, 2003.

A The CRSW Hash Function

In this section we recall the hash function construction of [CRSW13], a high level description appears in the introduction. Fix $n \leq m \leq u$ such that m is a power of two and $m = \alpha n$ for some constant $\alpha > 1$.

Construction A.1 (Variant of [CRSW13]). *Let $d \stackrel{\text{def}}{=} \Theta(\log\log(m))$ and $\delta = 1/\mathsf{poly}(m)$. For every $i \in [d]$, let $\mathcal{H}_i$ be a family of k_i-wise δ_i-dependent functions from $[u]$ to $\{0,1\}^{\ell_i}$, where:*

- *For every $i \in [d-1]$ let $\ell_i \stackrel{\text{def}}{=} \lfloor (1-\lambda)\lambda^{i-1} \cdot \log(m) \rfloor$, where $\lambda \in (0.5, 1)$ is a constant, and let $\ell_d \stackrel{\text{def}}{=} \log(m) - \sum_{i=1}^{d-1} \ell_i$. We set d to be such that $\ell_d = \Theta(\log\log(m))$.*
- *For every $i \in [d-1]$ let $k_i \stackrel{\text{def}}{=} \Theta\left(\frac{\log(m)}{\ell_i}\right)$, and let $k_d \stackrel{\text{def}}{=} \Theta(\log(m))$. We assume that k_i is even for all $i \in [d]$.*
- *For every $i \in [d-1]$ let $\delta_i \stackrel{\text{def}}{=} \delta$, and let $\delta_d \stackrel{\text{def}}{=} m^{-\Theta(\log\log(m))} \ll \delta$.*

We define the hash function family $\mathcal{H}_{\mathsf{CRSW}} = \{h : [u] \to \{0,\dots,m-1\}\}$ as the direct product of $\mathcal{H}_1,\dots,\mathcal{H}_d$; namely, the collection of all functions $h : [u] \to \{0,\dots,m-1\}$ of the form $h(x) = (h_1(x),\dots,h_d(x))$ where $h_1 \in \mathcal{H}_1,\dots,h_d \in \mathcal{H}_d$.

By using the hash functions of Theorem 2.2 we can instantiate Construction A.1 with a total seed length of $O(\log(m)\log\log(m))$. Using the fast evaluation of [MRRR13], we can also obtain $O((\log\log m)^2)$ evaluation time.

The following lemma, which is a close variant of [CRSW13, Lemma 3.2], shows that Construction A.1 is "load-balancing" at every level of granularity. That is, when hashing n items, the number of items that have a certain prefix of length k is very close to the expectation, $n/2^k$.

Lemma A.2 (c.f. [CRSW13, Lemma 3.2]). *For every constants $c_1, \epsilon > 0$ there exists a setting of parameters for $\mathcal{H}_{\mathsf{CRSW}}$ such that the following holds: for every subset $S \subseteq [u]$ of size n and for every string $z \in \{0,1\}^k$, where $k = \sum_{j=1}^{j^*} \ell_j$ for some $j^* \in \{0,\dots,d-1\}$:*

$$\Pr_{h \in_R \mathcal{H}_{\mathsf{CRSW}}}\left[\left|S \cap \bigcup_{i:\, pref_k(i)=z} h^{-1}(i)\right| \in (1 \pm \epsilon)\cdot\frac{|S|}{2^k}\right] > 1 - n^{-c_1}.$$

Proof (of Lemma A.2). For simplicity, and without loss of generality, we assume that z is the all zeros string (i.e., $z = 0^k$). Let $S_0 \overset{\text{def}}{=} S$ and for every $j \in \{1,\dots,j^*\}$, let $S_j \overset{\text{def}}{=} \{x \in S_{j-1} \colon h_j(x) = 0^{\ell_j}\}$. Fix $\gamma = \Theta(1/\log\log(m))$. The lemma follows from Claim A.3, with respect to $j = j^*$, and a suitable setting of parameters for $\mathcal{H}_{\mathsf{CRSW}}$ (so that $(1+\gamma)^d \le 1+\epsilon$ and $(1-\gamma)^d \ge 1-\epsilon$).

Claim A.3. *For every $j \in \{0,\dots,j^*\}$,*

$$\Pr_{h \in_R \mathcal{H}_{\mathsf{CRSW}}}\left[(1-\gamma)^j \frac{n}{2^{\sum_{i=1}^{j}\ell_i}} \le |S_j| \le (1+\gamma)^j \frac{n}{2^{\sum_{i=1}^{j}\ell_i}}\right] \ge 1 - \frac{j}{n^{2c_1}}$$

Proof. We prove the lemma by induction on j. If $j = 0$ the claim holds trivially. Suppose that the claim holds for some $j \in \{0,\dots,j^*-1\}$. By definition, S_{j+1} is a sum of $|S_j|$ indicator random variables. Moreover, since h_{j+1} is k_{j+1}-wise δ-dependent, these indicator variables are k_{j+1}-wise δ-dependent and S_{j+1} has expectation $\mu \overset{\text{def}}{=} |S_j|/2^{\ell_{j+1}}$. Hence, by Lemma 2.3,

$$\Pr\left[|S_{j+1}| \notin (1\pm\gamma)\mu\right] \le 2\left(\frac{|S_j|k_{j+1}}{(\gamma\mu)^2}\right)^{k_{j+1}/2} + \delta\left(\frac{|S_j|}{\gamma\mu}\right)^{k_{j+1}} = 2\left(\frac{2^{2\ell_{j+1}}k_{j+1}}{\gamma^2|S_j|}\right)^{k_{j+1}/2} + \delta\left(\frac{2^{\ell_{j+1}}}{\gamma}\right)^{k_{j+1}}$$

We bound each term separately. For the first term note that by the inductive hypothesis, with probability $1 - j/n^{2c_1}$ it holds that $|S_j| \ge (1-\gamma)^j \frac{n}{2^{\sum_{i=1}^{j}\ell_j}}$, in which case, by our setting of $\lambda > 0.5$ it holds that $|S_j| \ge (1-\gamma)^j 2^{\frac{1}{1+\lambda}\ell_{j+1}}$ and therefore

$$\left(\frac{2^{2\ell_{j+1}}k_{j+1}}{\gamma^2|S_j|}\right)^{k_{j+1}/2} \le 2^{-\Omega(\ell_{j+1}k_{j+1})} \le n^{-2c_1}/2$$

where the last inequality holds for a suitable setting of the parameters of Construction A.1. For the second term, by setting δ to be sufficiently small, we obtain:

$$\delta \left(\frac{2^{\ell_{j+1}}}{\gamma}\right)^{k_{j+1}} \leq n^{-2c_1}/2$$

and the claim follows. □

This concludes the proof of Lemma A.2. □

B Missing proof from Section 3

In this section we prove Theorems 3.2 to 3.4. In Section B.1 we prove Theorems 3.2 and 3.3 and in Section B.2 we prove Theorem 3.4.

B.1 Proving Theorems 3.2 and 3.3

We proceed to the actual proof. Fix constants $c_1, c_2, c_3 > 0$ as in the theorem statements, let $c_4 > 0$ be some sufficiently large constant (whose value, as a function of c_1, c_2 and c_3 will be determined below) and fix constants for Construction A.1 whose value (as a function of c_1, c_2, c_3 and c_4) will be determined below. For every $t \in [c_3 \log n]$, let $T_t \overset{\text{def}}{=} 2^{\sum_{i=j_t+1}^{d} \ell_i}$ for $j_t \in \{0, \dots, d-1\}$ which is defined as follows:

1. If $t < c_4$ (i.e., $t = O(1)$) then $j_t = 0$.
2. If[5] $(et)^{(1+c_2)t} \leq m$ (i.e., roughly $t = O(\log(n)/\log\log(n))$) then $j_t \overset{\text{def}}{=} \left\lfloor \frac{\log(t/c_4)}{\log(1/\lambda)} \right\rfloor$.
3. Otherwise, $j_t \overset{\text{def}}{=} d - 1$.

Jumping ahead, we note that t will denote the size of some "illegal" connected graph (i.e., a graph that is either too large or too dense) whose existence we need to rule out.

For every $h \in \mathcal{H}_{\mathsf{CRSW}}$, we denote by $h_1^{(t)}$ the $(\log(m) - \log(T_t))$-bit prefix of h and by $h_2^{(t)}$ the $\log(T_t)$-bit suffix of h. Note that $h = h_1^{(t)} \circ h_2^{(t)}$ and that, by construction of $\mathcal{H}_{\mathsf{CRSW}}$, if $h \in_R \mathcal{H}_{\mathsf{CRSW}}$ then $h_1^{(t)}$ and $h_2^{(t)}$ are independent (as random variables).

For every $t \in [c_3 \log n]$, we can view the graph $G_{h,g}(S)$ as being composed of blocks of T_t consecutive vertices. That is, the graph with vertex set m/T_t and an edge $(\lfloor u/T_t \rfloor, \lfloor v/T_t \rfloor)$ for every edge (u, v) in $G_{h,g}(S)$. Fix a constant $\beta \in (0, 1 - 1/\alpha)$. We say that the functions $(h_1^{(t)}, g_1^{(t)})$ are *t-good* if there are at most $(1 - \beta)\, T_t$ edges connected to each block of size T_t of $G_{h,g}(S)$.

Claim B.1. *Let $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$. For every $t \in [c_3 \log n]$, the functions $(h_1^{(t)}, g_1^{(t)})$ are t-good with probability $1 - n^{-2c_1}$, where the probability is only over $h_1^{(t)}$ and $g_1^{(t)}$ (and not over $h_2^{(t)}$ and $g_2^{(t)}$).*

[5] Here and throughout this work, e denotes the base of the natural logarithm.

Proof. Consider the i-th block of $G_{h,g}(S)$ (i.e., the block corresponding to vertices $T_t \cdot (i-1), \ldots, T_t \cdot i - 1$ of $G_{h,g}(S)$). The edges that are connected to vertices in this block correspond to values $x \in S$ such that either $h_1^{(t)}(x) = i$ or $g_1^{(t)}(x) = i$. By Lemma A.2 with respect to $\epsilon = \alpha \cdot (1-\beta) - 1 > 0$ (and using the fact that $T_t \geq 2^{\ell_d}$) and an appropriate instantiation of parameters of $\mathcal{H}_{\mathsf{CRSW}}$, with probability $1 - n^{-(2c_1+1)}$, the number of such x's is at most $(1+\epsilon)\frac{n}{2^{\log(m/T_t)}} = (1-\beta)\,T_t$. The claim follows by a union bound on all blocks.

This concludes the proof of Claim B.1 $\square$

Claim B.2. *For every $t \in [c_3 \log n]$, the hash functions $h_2^{(t)}$ and $g_2^{(t)}$ are $((1+c_2)t)$-wise δ_{j_t+1}-dependent.*

Proof. The $\log(T_t)$-bit suffix of $\mathcal{H}_{\mathsf{CRSW}}$ contains only output bits of the functions $h_{j_t+1}, \ldots, h_d$ which are all (independent) k_{j_t+1}-wise δ-dependent hash functions (note that $k_i \geq k_j$ for all $i \geq j$). We consider two cases:

– If $t < c_4$, then $j_t = 0$. Hence, $k_{j_t+1} = O(1)$ which is greater than $(1+c_2)c_4$ for a suitable choice of parameters for $\mathcal{H}_{\mathsf{CRSW}}$.
– If $(et)^{(1+c_2)t} \leq m$. In this case $j_t = \left\lfloor \frac{\log(t/c_4)}{\log(1/\lambda)} \right\rfloor$. Hence, $k_{j_t+1} = \Theta\left(\frac{\log(m)}{\ell_{j_t+1}}\right) = \Theta(\lambda^{-j_t}) = \Theta(t/c_4)$ which is greater than $(1+c_2)t$ for a suitable choice of parameters for $\mathcal{H}_{\mathsf{CRSW}}$.
– Otherwise (i.e., if $(et)^{(1+c_2)t} > m$) it holds that $j_t = d-1$. By Construction A.1, the ℓ_d-bit suffix of h and g (which simply corresponds to the last hash function in the construction) is k_d-wise δ_d-dependent, where $k_d = \Theta(\log(m))$ which is greater than $(1+c_2)t$ (for a suitable choice of parameters for $\mathcal{H}_{\mathsf{CRSW}}$).

This concludes the proof of Claim B.2. $\square$

The following lemma is the key lemma for the proofs of Theorem 3.2 and Theorem 3.3. Loosely speaking, it bounds the probability that a particular connected graph (V, E) appears. Note that this bound is meaningful when either $|V|$ is large (i.e., the graph is big) or $|E| \gg |V|$ is large (i.e., the graph is dense).

Lemma B.3. *Fix $x \in S$ and let $t \in [c_3 \log n]$. Let (V, E) be a connected graph such that $|V| = t$ and $|E| \leq (1+c_2)t$. Let $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$ conditioned on $(h_1^{(t)}, g_1^{(t)})$ being t-good. The probability that $G_{h,g}(S)$ contains (V, E) as a subgraph that contains the edge $(h(x), g(x))$ is at most*

$$\Big((1-\beta) \cdot T_t\Big)^{|E|-1} \cdot \left(\frac{1}{T_t^{2|E|-|V|}} + \delta_{j_t+1}\right)$$

where the probability is only over $h_2^{(t)}$ and $g_2^{(t)}$ (and not over $h_1^{(t)}$ and $g_1^{(t)}$).

Proof. Let $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$ conditioned on $(h_1^{(t)}, g_1^{(t)})$ being t-good.

We say that an injective function $\pi : E \to S$ is a *subgraph association* of (V, E) to $G_{h,g}(S)$ if π preserves the graph structure. Namely, for every pair of edges $(u, v), (v, w) \in E$ such that $\pi((u, v)) = y$ and $\pi((v, w)) = z$, for some $y, z \in S$, it holds that either $h(y) = h(z)$ or $h(y) = g(z)$ or $g(y) = h(z)$ or $g(y) = g(z)$.

We say that π is a *block association* if π preserves the structure at the block level (but not necessarily within the blocks). That is, for every $(u, v), (v, w) \in E$ such that $\pi(u, v) = x$ and $\pi(v, w) = y$ it holds that one of the followings pairs are in the same block (of size T_t) of $G_{h,g}(S)$: either $(h(y), h(z))$ or $(h(y), g(z))$ or $(g(y), h(z))$ or $((g(y), g(z))$. We note that every subgraph association is, in particular, a block association.

If the graph $G_{h,g}(S)$ contains (V, E) as a subgraph (that contains $(h(x), g(x))$), then there must exist a subgraph association of (V, E) to $G_{h,g}(S)$ that contains x (i.e., some edge in E is mapped to x). Hence, it suffices to upper bound the probability that such a subgraph association exists. The latter is done by first upper bounding the probability that any particular *block association* is a *subgraph association* and then applying a union bound over all possible block associations (that contain x).

We proceed to bound the probability that any particular block association is a subgraph association. Toward this end, note that by Claim B.2, $h_2^{(t)}$ and $g_2^{(t)}$ are $((1 + c_2)t)$-wise δ_{j_t+1}-dependent and that $|E| \leq (1 + c_2)t$. Hence, we can view the mapping within the blocks as being an (almost) totally random function.

Let $\pi : E \to S$ be some block association. Let v be a vertex and let $u_1, \ldots, u_d$ be the neighbors of v in (V, E). Since each block is of size T_t, the probability that all endpoints corresponding to v are mapped to the same value (in $[m]$) by π is at most $T_t^{d-1} + \delta_{j_t+1}$. Applying the same reasoning for every $v \in V$, we obtain that the probability that π is a subgraph association is at most

$$T_t^{-\sum_{v \in V}(d_v - 1)} + \delta_{j_t+1} = T_t^{|V| - 2|E|} + \delta_{j_t+1}$$

We proceed to bound the number of possible block associations from (V, E) to S that contain x. Let $(u, v) \in E$ be the edge that is mapped to x (which corresponds to the edge $(h(x), g(x))$ in $G_{h,g}(S)$). We consider an edge that is connected to (u, v) and without loss of generality assume that it is connected to v. That is, we assume that $(v, w) \in E$ for some $w \in V$. By our assumption that $(h_1^{(t)}, g_1^{(t)})$ is good, the number of edges connected to each block of $G_{h,g}(S)$ is at most $(1 - \beta)T_t$. Hence, the number of possibilities for mapping (v, w) to edges in $G_{h,g}(S)$ is at most $(1 - \beta)T_t$ (i.e., the number of edges connected to the block that contains v). We proceed like this iteratively, where at each stage we add an edge and increase the number of block associations by a factor of $(1 - \beta)T_t$. Hence, the total number of block associations that contain $h(x)$ is at most $((1 - \beta)T_t)^{|E|-1}$. The lemma follows by an application of the union bound (over all block associations that contain x).

This concludes the proof of Lemma B.3. □

Proof of Theorem 3.2. To prove Theorem 3.2 we use the union bound, over all (unlabeled) trees of size $c_3 \log n$, together with Lemma B.3 to bound the probability that a tree of size $c_3 \log n$ appears in $G_{h,g}(S)$. If no such tree appears, then clearly no connected component of size $c_3 \log n$ can appear. The proof follows.

Let (V, E) be a tree on $c_3 \log n$ vertices. By Lemma B.3, with respect to $t = |V| = c_3 \log n$ (note that $|E| = t - 1 < (1 + c_2)t$ and that $(et)^{(1+c_2)t} > m$), conditioned on $(h_1^{(t)}, g_1^{(t)})$ being t-good, for every $x \in S$, the probability that (V, E) is contained as a subgraph in $G_{h,g}(S)$ that contains the edge $(h(x), g(x))$ is at most

$$\Big((1-\beta)\cdot T\Big)^{|E|-1} \cdot \left(\frac{1}{T^{2|E|-|V|}} + \delta_d\right) \le \big(1-\beta\big)^{c_3 \log n - 2} + (2^{\ell_d})^{c_3 \log n - 2}\delta_d$$

Recall that by Lemma 2.4, the number of unlabeled trees of size $c_3 \log n$ is at most $4^{c_3 \log n}$. Hence, by the union bound (over all trees and all $x \in S$), the probability that $G_{h,g}(S)$ contains some tree (and therefore some connected subgraph) of size $c_3 \log n$ is at most

$$n \cdot 4^{c_3 \log n} \cdot \Big(\big(1-\beta\big)^{c_3 \log n - 2} + (2^{\ell_d})^{c_3 \log n - 2}\delta_d\Big) \le n^{-c_1} - n^{-2c_1}$$

where the last inequality follows by setting β (and hence also α) to be sufficiently large, $\delta_d = m^{-\Theta(\log\log(m))}$ to be sufficiently small and c_3 to be sufficiently large.

The lemma follows by noting that by Claim B.1 the functions (h_1, g_1) are t-good with probability $1 - n^{-2c_1}$.

Proof of Theorem 3.3 and Variants. In this section we prove Theorem 3.3 as well as some useful variants. As in the proof of Theorem 3.2 the main idea is to use Lemma B.3 and take a union bound over all dense graphs. The following claim bounds the probability for a dense subgraph of some particular size.

Claim B.4. *For every $t \in [c_3 \log n]$, the probability that $G_{h,g}(S)$ contains a connected subgraph (V, E) such that $|V| = t$ and $|E| = (1 + c_2)t$ is at most n^{-2c_1} for $t \ge c_4$ and is at most $O(1/n)$ for $t < c_4$.*

Proof. Let $t \in [c_3 \log n]$. By Claim B.1 we can restrict our attention to the case that (h_1, g_1) are t-good.

Let (V, E) be a connected graph such that $|V| = t$ and $|E| = (1 + c_2)t$. By Lemma B.3, the probability that (V, E) is a subgraph of $G_{h,g}(S)$, conditioned on (h_1, g_1) being t-good, is at most

$$n \cdot \Big((1-\beta)\cdot T_t\Big)^{|E|-1} \cdot \left(\frac{1}{T_t^{2|E|-|V|}} + \delta_{j_t+1}\right) \le n \cdot \frac{1}{T_t^{c_2 t+1}} + n \cdot T_t^{|E|}\delta_{j_t+1}$$

Since there are at most $\binom{|V|^2}{|E|} \le \left(\frac{et^2}{(1+c_2)t}\right)^{(1+c_2)t} \le (et)^{(1+c_2)t}$ such graphs, by the union bound, conditioned on $(h_1^{(t)}, g_1^{(t)})$ being t-good, the probability that $G_{h,g}(S)$ contains a connected subgraph (V, E) such that $|V| = t$ and $|E| = (1 + c_2)t$ is at most

$$n \cdot \frac{(et)^{(1+c_2)t}}{T_t^{c_2 t+1}} + n \cdot (et)^{(1+c_2)t}(T_t)^{(1+c_2)t}\delta_j.$$

We separate into cases:

1. If $t < c_4$, then $j_t = 0$ and therefore $T_t = m$. Hence,

$$n \cdot \frac{(et)^{(1+c_2)t}}{T_t^{c_2t+1}} + n \cdot (et)^{(1+c_2)t}(T_t)^{(1+c_2)t}\delta_j = n \cdot \frac{(et)^{(1+c_2)t}}{m^{c_2t+1}} + n \cdot (et \cdot m)^{(1+c_2)t}\delta$$
$$\leq \gamma/n$$

   for some constant $\gamma > 0$, where the last inequality follows by noting that $t < c_4$, that $c_2t = |E| - |V| \geq 1$ and by setting $\delta = 1/\mathsf{poly}(n)$ to be sufficiently small.
2. If $(et)^{(1+c_2)t} \leq m$, then $j_t = \left\lfloor \frac{\log(t/c_4)}{\log(1/\lambda)} \right\rfloor$ and therefore:

$$\log(T_t) = \log m - \sum_{i=1}^{j_t} \ell_i \geq \lambda^{j_t} \cdot \log(m) - d \geq \lambda \cdot \frac{c_4 \log(m)}{t} - d.$$

   and

$$\log(T_t) = \log m - \sum_{i=1}^{j_t} \ell_i \leq \lambda^{j_t} \cdot \log(m) \leq \frac{c_4 \log(m)}{t}.$$

   and therefore:

$$\begin{aligned} n \cdot \frac{(et)^{(1+c_2)t}}{T_t^{c_2t+1}} + n \cdot (et)^{(1+c_2)t}(T_t)^{(1+c_2)t}\delta &\leq \frac{nm}{T_t^{c_2t}} + nm \cdot (T_t)^{(1+c_2)t}\delta \\ &\leq \frac{nm}{2^{c_2c_4\lambda\log(m) - c_2td}} + nm^{c_4(1+c_2)+1} \cdot \delta \\ &\leq n^{-c_1}/4 \end{aligned}$$

   where the last inequality follows by letting c_4 be sufficiently large, letting $\delta = 1/\mathsf{poly}(n)$ be sufficiently small, and noting that $d = O(\log\log m)$ and that $t = O(\log(n)/\log\log(n))$ (since $(et)^{(1+c_2)t} \leq m$).
3. If $(et)^{(1+c_2)t} > m$, then $j = d - 1$ and $T_t = 2^{\ell_d} > (et)^{\frac{2(1+c_2)}{c_2}}$ (where the inequality follows from the fact that $t \leq c_3 \log n$ and by setting $\ell_d = \Omega(\log\log(n))$ for a suitable constant in the Ω notation). Hence,

$$\begin{aligned} n \cdot \frac{(et)^{(1+c_2)t}}{T_t^{c_2t+1}} + n \cdot (et)^{(1+c_2)t}(T_t)^{(1+c_2)t}\delta_d &\leq n \cdot \frac{1}{(et)^{(1+c_2)t}} + n \cdot (et)^{(1+c_2)t}T^t\delta_d \\ &\leq n \cdot n^{-\Theta(\log\log(n))} + n^{\Theta(\log\log(n)))}\delta_d \\ &\leq n^{-2c_1} \end{aligned}$$

where the last inequality follows by setting $\delta_d = m^{-\Theta(\log\log(m))}$ to be sufficiently small.

This concludes the proof of Claim B.4. □

Using Claim B.4 we can easily prove Theorem 3.3.

Proof (of Theorem 3.3). The theorem follows by applying the Claim B.4 for every $t \in [c_3 \log n]$ and taking a union bound. $\square$

Note that the probability for a bad event in Theorem 3.3 is inversely proportional to a fixed polynomial (specifically $O(1/n)$). For some applications (see Section 4.1) it is desirable to obtain probability that is inversely proportional to an *arbitrary* polynomial. Unfortunately, such a result cannot hold (even when using completely random functions) since the probability for a tiny (e.g., of size 1) but dense connected component is a fixed polynomial. Nevertheless, we prove three useful variants of Theorem 3.3 which do provide such a guarantee. In Theorem B.5, we consider only dense graphs of some minimal (constant) size. In Theorem B.6, we consider graphs with higher (constant) density and finally in Theorem B.7 we consider a collection of graphs.

Theorem B.5. *For every constant $c_1, c_2 > 0$ there exists a constant $t > 0$ and a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every $S \subseteq [u]$, with probability $1 - n^{-c_1}$ over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$, for every connected subgraph (V, E) of the graph $G_{h,g}(S)$ of size $|V| \geq t$ it holds that $|E| < (1 + c_2) \cdot |V|$.*

Proof. The theorem follows directly by applying Claim B.4 for every $t > c_4$, a union bound, and Theorem 3.2. $\square$

Theorem B.6. *For every constant $c_1 > 0$ there exists a constant $s > 0$ and a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every $S \subseteq [u]$, with probability $1 - n^{-c_1}$ over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$, for every connected subgraph (V, E) of the graph $G_{h,g}(S)$ it holds that $|E| < s \cdot |V|$.*

Proof. By Theorem B.5 it suffices to prove the theorem for subgraphs of size at most c_4. Let $t < c_4$ and let s be a sufficiently large constant. Let (V, E) be a connected graph such that $|V| = t$ and $|E| = s \cdot t$. By Lemma B.3, the probability that (V, E) is a subgraph of $G_{h,g}(S)$, conditioned on (h_1, g_1) being t-good, is at most

$$n \cdot \Big((1 - \beta) \cdot T_t \Big)^{|E|-1} \cdot \left(\frac{1}{T_t^{2|E|-|V|}} + \delta_{j_t+1} \right) \leq n \cdot \frac{1}{T_t^{(s-1)t}} + n \cdot T_t^{|E|} \delta_{j_t+1}$$

Since there are at most $\binom{|V|^2}{|E|} \leq \left(\frac{et^2}{st}\right)^{st} \leq t^{st}$ such graphs, by the union bound, conditioned on $(h_1^{(t)}, g_1^{(t)})$ being t-good, the probability that $G_{h,g}(S)$ contains a connected subgraph (V, E) such that $|V| = t$ and $|E| = s \cdot t$ is at most

$$n \cdot \frac{t^{st}}{T_t^{(s-1)t}} + n \cdot t^{st} \cdot T_t^{|E|} \delta_{j_t+1}$$

Since $t < c_4$, then $j_t = 0$ and therefore $T_t = m$. Hence,

$$n \cdot \frac{t^{st}}{T_t^{(s-1)t}} + n \cdot t^{st} \cdot T_t^{|E|} \delta_{j_t+1} \leq n \cdot \frac{c_4^{(c_4 s)}}{m^{s-1}} + n \cdot c_4^{(c_4 s)} \cdot m^{c_4 s} \delta \leq n^{-c_1}$$

where the last inequality follows by setting s to be sufficiently large and $\delta = 1/\mathsf{poly}(n)$ to be sufficiently small. $\square$

Theorem B.7. *For every constants $c_1, c_2 > 0$ there exists a constant $t > 0$ and a suitable setting of constants for $\mathcal{H}_{\mathsf{CRSW}}$ such that for every $S \subseteq [u]$, the probability, over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$, that $G_{h,g}(S)$ contains a collection of edge disjoint connected subgraphs $\{(E_1, V_1), \dots, (E_k, V_k)\}$ such that $|E_i| \geq (1 + c_2)|V_i|$ for every $i \in [k]$, and $\sum_{i\in[k]} |V_i| = t$, is at most n^{-c_1}.*

Proof. Let t be a sufficiently large constant and fix the parameters of $\mathcal{H}_{\mathsf{CRSW}}$ so that $k_1 \geq (1 + c_2)t$ (in particular, the functions $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$ are $(1 + c_2)t$ wise δ-dependent.

Fix a collection of connected graphs $\{(E_1, V_1), \dots, (E_k, V_k)\}$ such that $|E_i| = (1 + c_2)|V_i|$ for every $i \in [k]$, and $\sum_{i\in[k]} |V_i| = t$. Since the functions h and g are $\sum_{i=1}^{k} |E|_i$-wise δ-dependent, the probability that $G_{h,g}(S)$ contains $\{(E_1, V_1), \dots, (E_k, V_k)\}$ as edge disjoint subgraphs is at most:

$$\left(\binom{n}{|V_1|} m^{-(|E_1|-1)}\right) \cdot \left(\binom{n - |V_1|}{|V_2|} m^{-(|E_2|-1)}\right) \cdot \dots \cdot \left(\binom{n - \sum_{i=1}^{k-1} |V_i|}{|V_k|} m^{-(|E_k|-1)}\right) + \delta$$

which is at most $n^{\sum_i |V_i|} \cdot m^{-\sum_i |E_i|} + \delta \leq n^{-(c_1+1)}$, where the inequality follows from the fact that $|E_i| = (1 + c_2)|V_i|$ and setting t to be sufficiently large and δ to be sufficiently small. The theorem follows by taking a union bound over at most

$$\prod_{i=1}^{k} \binom{|V_i|^2}{|E_i|} = O(1)$$

such collections. $\square$

B.2 Connected Components have Expected Constant Size

Theorem 3.2 shows that with high probability every connected component has at most logarithmic size. In this section we complement this result by proving Theorem 3.4 that shows that in *expectation* each connected component has constant size. To prove the lemma, intuitively, we show that the size of the connected component is distributed (roughly) geometrically.

Proof (of Theorem 3.4). Fix $x \in S$. To prove Theorem 3.4 we first prove the following claim.

Claim B.8. *For every $t \in [c_3 \log n]$, the probability, over the choice of $h, g \in_R \mathcal{H}_{\mathsf{CRSW}}$, that the edge $(h(x), g(x))$ is contained in a connected subgraph of size t is at most $c\gamma^t + n^{-3}$, for some constants $\gamma \in (0, 1)$ and $c > 0$.*

Proof. Let (V,E) be a tree on t vertices. By Lemma B.3, conditioned on $h_1^{(t)}$ and $g_1^{(t)}$ being t-good, the probability that (V,E) appears as a subgraph that contains the edge $(h(x),g(x))$ in $G_{h,g}(S)$ is at most $(1-\beta)^{t-2}+(T_t)^t\delta_{j_t+1}$.

By Lemma 2.4, the number of trees of size t is at most 4^t. Hence, by the union bound, the probability that the edge $(h(x),g(x))$ is contained in some tree (and therefore in any connected subgraph) of size t, conditioned on $h_1^{(t)}$ and $g_1^{(t)}$ being t-good is at most:

$$4^t\left((1-\beta)^{t-2}+(T_t)^t\delta_{j_t+1}\right)=(1-\beta)^{-2}(4(1-\beta))^t+(4T_t)^t\delta_{j_t+1}\le c\gamma^t+n^{-4}$$

where $\gamma\in(0,1)$ and $c>0$ are constants and the last inequality follows by setting β (and hence also α) to be sufficiently large and δ_{j_t+1} to be sufficiently small. The claim follows by Claim B.1.

This concludes the proof of Claim B.8. □

Hence, the expected size of the connected component that the edge $e_x \overset{\text{def}}{=} (h(x),g(x))$ belongs to is

$$\begin{aligned}\sum_{t=1}^{n} t\cdot\Pr[e_x \text{ is in a connected graph of size } t] &\le \sum_{t=1}^{c_3\log n} t\cdot\Pr[e_x \text{ is in a connected graph of size } t]\\ &\quad + n^2\Pr[e_x \text{ is in connected graph of size } c_3\log n]\\ &\le \sum_{t=1}^{c_3\log n} t\cdot(c\gamma^t+n^{-3})+n^2(c\gamma^{-c_3\log n}+n^{-3})\\ &= O(1)\end{aligned}$$

where the last equation follows by setting c_3 to be sufficiently large. □