Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Simple Set Sketching

Jakob Bæk Tejs Houen* Rasmus Pagh† Stefan Walzer‡

Abstract

Imagine handling collisions in a hash table by storing, in each cell, the bit-wise exclusive-or of the set of keys hashing there. This appears to be a terrible idea: For αn keys and n buckets, where α is constant, we expect that a constant fraction of the keys will be unrecoverable due to collisions.

We show that if this collision resolution strategy is repeated *three times* independently the situation reverses: If α is below a threshold of ≈ 0.81 then we can recover the set of all inserted keys in linear time with high probability.

Even though the description of our data structure is simple, its analysis is nontrivial. Our approach can be seen as a variant of the *Invertible Bloom Filter* (IBF) of Eppstein and Goodrich. While IBFs involve an explicit checksum per bucket to decide whether the bucket stores a single key, we exploit the idea of quotienting, namely that some bits of the key are implicit in the location where it is stored. We let those serve as an implicit checksum. These bits are not quite enough to ensure that no errors occur and the main technical challenge is to show that decoding can recover from these errors.

1 Introduction

Sketching is the idea of representing data in a compact, potentially lossy form. For this introduction imagine that, for some sets $\mathcal{X}$ and $\mathcal{Y}$, a long (typically sparse) sequence $X \in \mathcal{X}^u$ is represented via a short sequence $f(X) \in \mathcal{Y}^n$ — the sketch of X — where $n \ll u$ and f is a (possibly randomized) function. We speak of *linear sketching* when $(\mathcal{X}, \oplus)$ and $(\mathcal{Y}, \oplus)$ are groups and f is a linear function, i.e. when $f(X \oplus X') = f(X) \oplus f(X')$ holds (component-wise) for all X, X'.

Linear sketches of data have appealing properties for applications in streaming or distributed settings [Woo14]. In particular, such sketches can be merged/updated to form a sketch of the combined data. This paper considers the case of $\mathcal{X} = \{0,1\}$, meaning the input $X \in \{0,1\}^u$ is conceptually a set $S \subseteq [u] := \{1, \ldots, u\}$ of *keys*. We assume that $u+1$ is a power of 2.

We present a new extremely simple approach for linear sketching of sets. It uses $\mathcal{Y} = \{0, \ldots, u\}$, hence an array $A \in \{0, \ldots, u\}^n$ where n is the selected size of the sketch, as well as independent hash functions $h_1, h_2, h_3 : [u] \to [n]$. Given a sketch A of $S \subseteq [u]$ we can add $x \notin S$ to the sketch (i.e. obtain a sketch of $S \cup \{x\}$) by setting $A[i] \leftarrow A[i] \oplus x$ for $i = h_1(x), h_2(x), h_3(x)$, where $\oplus$ denotes bit-wise exclusive-or.

This is indeed a linear sketch if addition in $\mathcal{X} = \{0,1\}$ and $\mathcal{Y} = \{0, \ldots, u\}$ are both understood to be bit-wise exclusive-or. Merging sketches of sets S_1 and S_2 will produce a sketch of the symmetric difference $S_1 \triangle S_2$. As long as there is only one copy of each element in the sets represented by the sketches we merge, we get a sketch of the union. We will see that from a sketch of a set S with $n \geq 1.23\,|S|$ we can recover S with high probability in linear time.

A simple scenario where this is useful is that of *set reconciliation* [MTZ03], where two parties, Alice and Bob, have sets S_1 and S_2 with a large overlap, and want to compute the union $S_1 \cup S_2$. If Alice computes a sketch of S_1 and sends it to Bob, he will be able to compute the sketch of $S_1 \triangle S_2$. If $n \geq 1.23\,|S_1 \triangle S_2|$ then Bob can recover $S_1 \triangle S_2$ and hence $S_1 \cup S_2$ with high probability. Remarkably, the size n of the sketches and hence the amount of information to be transferred is linear in $|S_1 \triangle S_2|$ rather than being linear in $|S_1 \cup S_2|$ and therefore close to the information-theoretical lower bound, which holds even if Alice knows which of her elements Bob is missing.

There is a rich literature on streaming algorithms (see e.g. the surveys [CJ19, McG14, Woo14]). Most streaming algorithms are linear sketches over the reals or integers, i.e. with $\mathcal{X} = \mathbb{Z}$ or $\mathcal{X} = \mathbb{R}$. Linear sketches over finite fields like considered in this paper are less well-studied, but are natural in some applications. For

*BARC and University of Copenhagen. Supported by the VILLUM Foundation grant 16582
†BARC and University of Copenhagen. Supported by the VILLUM Foundation grant 16582
‡University of Cologne. Supported by DFG grant WA 5025/1-1.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

method	year	$\frac{\text{space}}{m}$	t_{update}	t_{decode}	techniques
randomized k-set structure [Gan07]	2005	$O(\log m)$	$O(\log m)$	$\mathcal{O}(m \log m)$	A R – – M
deterministic k-set structure [GM08]	2006	2	$2m$	$\tilde{\mathcal{O}}(m^3)$	A – – – M
symmetric polynomials [EG11]	2007	1	m	$\tilde{\mathcal{O}}(m^2)$	A – – – –
IBF [EG11]	2007	$\mathcal{O}(\log(m))$	$\mathcal{O}(\log(m))$	$\tilde{\mathcal{O}}(m)$	– R P C M
IBLT ($k = 3$) [GM11]	2011	3.666	9	$\mathcal{O}(m)$	– R P C M
$\langle$*this paper,* $k = 3\rangle$	2023	1.222	3	$\mathcal{O}(m)$	– R P – –

Figure 1: Comparison of linear sketches for sets and multisets as discussed in Section 1.1, normalised such that decoding is possible if the set size is at most m. The space-column counts how many entries have to be stored and t_{update} counts how many entries are touched by insertions and deletions. The last column indicates which approaches use **A**lgebraic techniques, **R**andomisation, **P**eeling and **C**hecksums, and which approaches support **M**ultisets. All randomized sketches have a failure probability of $\mathcal{O}(1/m)$.

example, consider "straggler identification" [EG11], where there is a stream of events of the form enter(x) and exit(x), for elements $x \in [u]$ (e.g. think about employees entering and leaving a building, or locks being held in a database system). We want to be able to keep track of which elements have an enter(x) event without a matching exit(x) event, assuming that the number of such elements is low (e.g. employees left in the building at the end of a working day). Similarly, for the set reconciliation problem mentioned above, working with a sketch over the field of size two works just as well as working over the integers.

1.1 Related work. We summarise related work in Figure 1. Each of the listed competitors is a linear set sketch that stores a set S of integers or elements of some finite field. The sketches support insertions and deletions of elements as well as a decode operation that can reproduce S whenever $|S| \leq m$ for some parameter m. For simplicity we measure required space by counting how many numbers have to be stored, regardless of whether these are from $\mathbb{Z}$, from a finite field or from $[\mathcal{O}(m)]$. Crucially, the space requirement of all sketches only depends on m, even though $|S|$ is unlimited. Note that even though decoding is impossible as long as $|S| > m$, it must become possible again if and when $|S| \leq m$ holds again at some later point.

The following ideas are shared by several of the listed approaches.

Multisets. Some approaches allow storing a multiplicity for each element in the set. Unsurprisingly, this tends to double the space requirement.

Algebraic techniques. A set $S \subset \mathbb{Z}$ of size m is uniquely determined by its power sums $(\sum_{x \in S} x^i)_{i \in [m]}$. This directly leads to a construction in [EG11] using symmetric polynomials and – less directly – to the k-set data structures in [GM08]. These approaches work deterministically, but have relatively slow update and decode operations.

Randomisation. A rather primitive linear sketch of a set S of group elements is the sum of the elements. Clearly, when $|S| \leq 1$ and S does not contain the neutral element then S can be recovered from the sketch.

All randomised approaches use a variant of such a primitive linear sketch in each of a large number of buckets. For each key hash functions select a small number of buckets in which the key is stored. During decoding the hope is that for every key x at least one of its buckets stores no key other than x, so that x can be recovered from this bucket.

For randomised approaches decoding may fail even though $|S| \leq m$. For better comparability we have tuned all competitors to have failure probabilities of $\mathcal{O}(1/m)$ in Figure 1.[1]

Two further techniques are often combined with randomised approaches:

Checksums. The decoding algorithm has to decide whether a value x stored in a bucket corresponds to the single key x or to the sum of several keys overlapping in the bucket. Both invertible Bloom filters (IBFs) [EG11] and invertible Bloom lookup tables (IBLTs) [GM11] use explicit hash checksums in each bucket to make this decision. A sanity check proposed in [EG11] that is central to our approach

[1] If an IBF [EG11] is tuned for failure probability ε, then space and update times are correspondingly reduced to $\mathcal{O}(\log(1/\varepsilon))$.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Algorithm initialise**:**
  $A[1,\dots,n] = (0,\dots,0)$

Algorithm toggle(x)**:**
  for $i \in h(x)$ **do**
    $A[i] \leftarrow A[i] \oplus x$

Algorithm merge($A' \in \{0,\dots,u\}^n$)**:**
  for $i \in [n]$ **do**
    $A[i] \leftarrow A[i] \oplus A'[i]$

Algorithm looksPure($i \in [n]$)**:**
  return $A[i] \neq 0 \wedge i \in h(A[i])$

Algorithm decode**:**
  $S_{\text{dec}} \leftarrow \varnothing$
  $Q \leftarrow \{i \in [n] \mid \text{looksPure}(i)\}$
  while $Q \neq \varnothing$ **do**
    $Q_{\text{next}} \leftarrow \varnothing$
    for $i \in Q$ **if** looksPure(i) **do**
      $x \leftarrow A[i]$ // *detected* key x
      toggle(x) // $S \leftarrow S \triangle \{x\}$
      $S_{\text{dec}} \leftarrow S_{\text{dec}} \triangle \{x\}$
      $Q_{\text{next}} \leftarrow Q_{\text{next}} \cup \{i \in h(x) \mid \text{looksPure}(i)\}$
    $Q \leftarrow Q_{\text{next}}$
  if $A[1,\dots,n] \neq (0,\dots,0)$ **then**
    return FAILURE
  return S_{dec} // correct whp

Figure 2: Implementation of simple set sketches.

is that x can only be stored in a bucket i, if i is one of the buckets selected for x by the hash functions. This check can act as an implicit checksum.

Peeling. Suppose that only a subset $S' \subset S$ of the elements in the sketch are directly recoverable due to being alone in a bucket. However, after removing S' from the sketch we obtain a sparser sketch where further elements may be recoverable. Peeling is the natural iterative decoding algorithm arising from the simple insight. It is used by IBFs (though not to its full potential), by IBLTs, and in this paper.

Our technical contribution is related to the work of Jiang, Mitzenmacher, and Thaler [JMT16], which studies parallel algorithms for peeling processes such as the one used in IBLTs. They show that only $\mathcal{O}(\log \log n)$ rounds of peeling are needed in a "breadth-first" peeling approach, similar to the one we use.

1.2 Contribution. We describe the *simple set sketch*, a randomised dynamic set data structure in the spirit of the IBF [EG11] and the IBLT [GM11]. At any point in time the sketch represents a set $S \subseteq [u]$ where $u = 2^w - 1$, i.e. keys are non-zero strings of w bits. Initially $S = \varnothing$. A toggle operation can be used to change the membership status of a given key $x \in [u]$, meaning that toggle(x) changes the represented set from S to $S \triangle \{x\}$ where $\triangle$ denotes the symmetric difference operator on sets. A merge operation takes another sketch representing a set S' as input and changes the represented set from S to $S \triangle S'$.

While no direct membership queries are supported, a decode operation tries to reconstruct the represented set S in its entirety, and succeeds with high probability under certain conditions discussed below.

The construction uses an array $A = A[1,\dots,n]$ of n buckets, each of which can store exactly one element of $\{0,\dots,u\}$, and a constant number $k \geq 3$ of uniformly random hash functions $h_1,\dots,h_k : [u] \to [n]$. We define $h(x) := \{h_1(x),\dots,h_k(x)\}$ as a multiset of size exactly k, noting that $h(x)$ is an ordinary set with probability $1 - \mathcal{O}(1/n)$.[2] The operations are implemented as shown in Figure 2.

The toggle-operations are commutative and two toggle(x) operations with the same x cancel. Hence, the state of the data structure is a function of h and the currently stored set S. Since A can assume at most u^n states while there are 2^u possibilities for S, the representation is necessarily "lossy" when S is large and n is small. Like regular IBLTs [GM11], decode relies on *peeling*, meaning we attempt to identify buckets i such that $A[i]$ is the trivial sum of just one key x and hence $A[i] = x$. We call such buckets *pure*. If detected, the key x is toggled – which removes it from the sketch – and x is recorded in the set S_{dec} to be returned in the end. A fully successful decode will leave the sketch empty.

To decide whether a bucket i is pure and stores a single key $A[i] = x$ or whether it stores a sum $A[i] = x_1 \oplus \ldots \oplus x_\ell$ of several keys, the looksPure function checks whether $A[i]$ hashes to i, i.e. whether $i \in h(A[i])$. This exploits that when $A[i]$ is single key then $i \in h(A[i])$ always holds, while if $A[i]$ is the sum of

[2] We could have forced $h_1,\dots,h_k$ to always produce distinct hashes and avoid multisets. However, then $h_1(x),\dots,h_k(x)$ would not be stochastically independent. So both choices involve mildly annoying (but ultimately inconsequential) technicalities.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

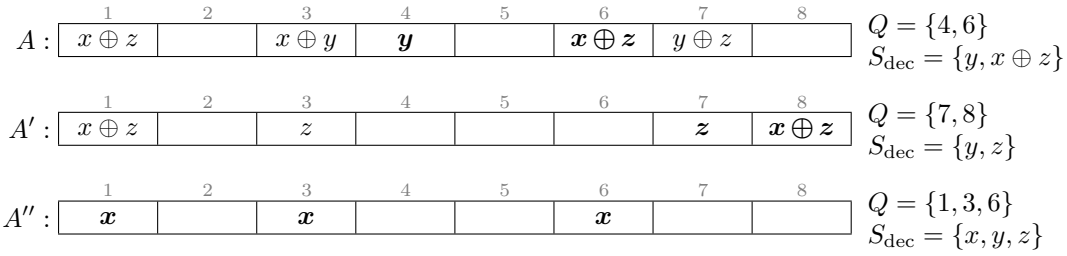

Figure 3: Example of simple set sketch decoding. The sketch A stores $S = \{x, y, z\}$, assuming $h(x) = \{1, 3, 6\}, h(y) = \{3, 4, 7\}, h(z) = \{1, 6, 7\}$. Moreover, assume $h(x \oplus z) = \{3, 6, 8\}$. Only y is alone in bucket 4, but bucket 6 with the foreign key $x \oplus z$ also looks pure, without actually being pure. In the first round we therefore have $Q = \{4, 6\}$ and toggle y and $x \oplus z$, resulting in A'. In the second round buckets $Q = \{7, 8\}$ look pure, with z and $x \oplus z$, so we toggle these keys and update the set of decoded keys to $S_{\text{dec}} = \{y, x \oplus z\} \triangle \{z, x \oplus z\} = \{y, z\}$. In the third and final round the remaining key x is recovered from A''.

several keys then $i \in h(A[i])$ is a coincidence, albeit one that does occur, as we will show later, an expected constant number of times overall. We leave decode oblivious of the issue of such *anomalies* and let it trust the output of looksPure. That way, it will sometimes erroneously detect a *foreign key*, z, that is not actually in the set[3]. The algorithm will try to remove z by calling toggle(z), but since z is not in the set, this will end up *adding* z to the data structure. If in the long run the ordinary decoding steps outnumber the anomalous decoding steps, i.e. when more keys are removed than added, then z will likely be isolated in a bucket at a later point. At this point, z will be toggled a second time, this time amounting to an actual removal from the sketch and from S_{dec}. This allows decode to rectify prior mistakes and return the correct set with high probability. The implementation uses two nested loops and we call an iteration of the outer loop a *round*. An example for the execution of decode is given in Figure 3. The main technical challenge will be to control the number and properties of anomalous decoding steps so that a successful recovery from the corresponding mistakes occurs with high probability.

In the following theorem the constant $c_k^{\triangle}$ is known as the *peeling threshold* or the threshold for the occurrence of a 2-core in a random k-uniform hypergraph. The largest, and hence most interesting of these values is $c_3^{\triangle} \approx 0.81$, relevant for $k = 3$ hash functions.

THEOREM 1.1. *Assume we have a sketch as explained above with n buckets and $k \geq 3$ hash functions representing a set S_0 of m keys where $\frac{m}{n} < c_k^{\triangle} - \varepsilon$ for some $\varepsilon > 0$. Then* decode *returns S_0 in time $\mathcal{O}(n)$ with high probability (whp, meaning with probability $1 - \tilde{O}(1/n)$).*

We remark that the error probability $\Theta(1/n)$ accounts for three ways in which decode can fail to return the correct set.

(1) decode may return FAILURE. This is a likely outcome when two keys $x, y \in S_0$ satisfy $h(x) = h(y)$, i.e. when they share all 3 hash values. Such keys exist with probability $\Theta(1/n)$.

(2) decode may fail to terminate. Assume for instance that $S_0 = \{1, 2\}$ with $h(1) = h(2) = \{a, b, c\}$ and $h(3) = \{c, d, e\}$ for some distinct buckets $a, b, c, d, e \in [n]$. The algorithm would erroneously select bucket c for decoding since $A[c] = 1 \oplus 2 = 3$ and $c \in h(3)$ – hence looksPure(c) is satisfied. This leads to key 3 being added to the sketch. Afterwards 3 is correctly detected to be the only key stored in bucket d (or e) and toggled a second time, bringing us back to the state we started in. A similar constellation of keys exists with probability $\Omega(1/n^2)$ in any set of $\Omega(n)$ keys.

(3) decode may return a set S_{dec} with $S_{\text{dec}} \neq S_0$. Assume for instance that for $S_0 = \{1, 2, 3\}$ we have $h(1) = h(2) = h(3)$, which happens with probability $\Omega(1/n^6)$. We then get $S_{\text{dec}} = \varnothing$ since the contributions $1 \oplus 2 \oplus 3 = 0$ cancel out everywhere.

The second and third failure cases are more problematic than the first. A practical implementation can prevent (2) by terminating the algorithm with "FAILURE" when it runs unexpectedly long. Moreover, it can reduce the probability of (3) to 2^{-r} by introducing a corresponding r-bit checksum, i.e. maintaining $C = \bigoplus_{x \in S} f(x)$

[3]Our notion of a foreign key has nothing to do with the notion of the same name used in the database literature.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

together with the sketch where $f : [u] \to \{0,1\}^r$ is a random hash function. Note that C is much more light-weight than the per-bucket checksums used in [EGUV11].

1.3 Technical Overview. From a high level, the analysis has four parts in corresponding subsections.

2.1 The Issue of Anomalies. We connect the "runtime" phenomenon of anomalous decoding steps to the "offline" combinatorial structure of *anomalies*. An anomaly is a set of keys $A = \{x_1, \dots, x_\ell\} \subseteq [u]$ with $x_1 \oplus \dots \oplus x_\ell = 0$ and a shared bucket $i \in h(x_1) \cap \dots \cap h(x_\ell)$. The presence of any $\ell - 1$ keys from A are, as far as the *centre* bucket i of A is concerned, indistinguishable from the presence of the missing ℓth key from A. This may cause i to lookPure, causing the missing key to be toggled and effectively added to the sketch. Every anomalous decoding step has such an underlying anomaly.

2.2 Isolating Anomalies. An anomaly A becomes relevant at runtime, as soon as $|A| - 1$ of its keys are present in the sketch. Initially only anomalies with at most one foreign key $x \in A \setminus S_0$ are relevant in this sense. We call such anomalies *native anomalies*. However, since native anomalies can cause foreign keys to be added to the sketch, anomalies with two or more foreign keys can become relevant as well, causing additional foreign keys to be added in an escalating cascade.

We show that no such cascade occurs whp. In fact, we show that only $\mathcal{O}(1)$ native anomalies exist in expectation (and $\mathcal{O}(\log n)$ whp) and that these take only the most harmless of forms with no mutual interaction. Concretely, native anomalies have a "star-shape", i.e. keys share only the centre bucket (formally $|h(A)| = (k-1)|A| + 1$) and any two native anomalies have disjoint domains ($h(A_1) \cap h(A_2) = \varnothing$).

2.3 Working Around Anomalies. Keys that are part of anomalies may be repeatedly toggled by decode, i.e. inserted and deleted many times. To obtain a clearer view on the lasting progress that is made, we consider a variant of decode where the dizzying commotion around anomalies is artificially frozen. More precisely, we let $S_{\mathcal{A}}$ be the set of *anomalous keys*, that is, the keys contained in native anomalies, and $B_{\mathcal{A}} = h(S_{\mathcal{A}})$ the set of *anomalous buckets*. We then consider a variant decode' of decode that is given $B_{\mathcal{A}}$ as an input and is banned from considering these buckets.

With the issue of anomalies out of the picture, decode' can be analysed with known techniques, which we postpone to Section 2.4. There we show that all buckets, except for those in $B_{\mathcal{A}}$, are cleared of keys whp. While decode may (repeatedly) remove and add keys disregarded by decode', we show that any key that is removed by decode' is also permanently removed by decode. From this we conclude that decode must reach a state where only anomalous keys are left. It is then not hard to see that these anomalous keys cannot survive in isolation. For each anomaly A and each remaining $x \in A$ there are $k - 1$ pure buckets only containing x, compared to only a single bucket (the centre of A) that could look pure without actually being pure. With such a majority of helpful over deceptive information, what is left of the anomaly will unravel within two rounds at most.

2.4 Analysis of decode'. We adapt the analysis of cores in hypergraphs by Molloy [Mol05] to our setting with anomalous buckets. A crucial lemma by Molloy [Mol05, Lemma 3] shows that only a constant fraction of hyperedges remain whp after a constant number of rounds when peeling a fully random k-uniform hypergraph. In our setting, this corresponds to only a constant fraction of the keys remaining after a constant number of iterations of the outer loop of decode if we have perfect information on which buckets are pure. We show that since there are only $\mathcal{O}(\log^2 n)$ anomalous buckets whp, which we block from consideration in decode', their effect on the peeling process cannot be too large, and we still obtain that only a constant fraction of the keys remain after a constant number of iterations of the outer loop of decode'.

We then employ a standard argument to show that if we have fewer than δn keys then at least a constant faction of these keys are isolated in a bucket whp. Now if we have n' isolated keys then we have at least $n' - |B_{\mathcal{A}}|$ buckets that are detected as pure by decode'. This shows that decode' will arrive at a point where at most $\Omega(|B_{\mathcal{A}}|)$ keys from $S \setminus S_{\mathcal{A}}$ are left. Finally, we need to argue that the last non-anomalous keys are also removed by decode', which is done by a technical counting lemma.

2 Analysis of the Decode Operation

2.1 The Issue of Anomalies. We begin by introducing concepts that will come in handy in the subsequent analysis.

We denote by $S_0 \subseteq [u]$ the set stored in the sketch before the decode operation is executed. When discussing states of the sketch while decode is in progress, S refers to the set of keys currently stored in the sketch and S_{dec} refers to the current state of the corresponding variable. Both S and S_{dec} may contain native keys, i.e. keys from S_0, as well as foreign keys, i.e. keys from $[u] \setminus S_0$. Since changes to S and S_{dec} happen in sync, $S_0 = S_{\text{dec}} \triangle S$ is an invariant of decode. It implies successful termination if and only if $S = \varnothing$ is reached.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Each iteration of the while loop carries out a *round* and each iteration of the for-loop where i *looks pure* (i.e. $\mathsf{looksPure}(i)$ holds) carries out a *step* at bucket i. We say the key $x = A[i]$ seemingly stored in bucket i is *detected* and toggled. If we in fact had $x \in S$, then S loses an element and we speak of a *regular step*, otherwise S gains an element and we speak of an *anomalous step*.

Anomalies. An anomalous step occurs when bucket i stores several elements $x_1, \dots, x_{\ell-1} \in [u] \setminus \{x\}$ with $x_1 \oplus \dots \oplus x_{\ell-1} = x$. An anomalous step is always linked to an *anomaly* of size ℓ.

DEFINITION 2.1. *A set $A = \{x_1, \dots, x_\ell\} \subseteq [u]$ with $\bigoplus_{j\in[\ell]} x_j = 0$ and $i \in h(x_j)$ for all $j \in [\ell]$ is an anomaly of size ℓ with centre $i \in [n]$.*[4]

An anomaly A of size ℓ is *triggered* if exactly $\ell - 1$ of its keys are stored in the sketch, i.e. if $A \cap S = A \setminus \{x\}$ for some $x \in A$, and no other key is stored in the centre bucket i. It then appears as though only x is stored in bucket i, i.e. i looks pure. An anomalous step may then detect key x in i and add x to S. Note that x may be native or foreign.

Native anomalies. Call an anomaly A a *native anomaly* if it contains at most one foreign key. A native anomaly may already be triggered when decoding starts, or can be triggered simply by removing keys stored in the centre of A. In principle, a *foreign anomaly*, i.e. an anomaly containing at least two foreign keys, can be triggered, provided that at least one of its keys is added to S during decoding due to different anomalies that are triggered prior to A. A non-trivial step in our argument is to show that only native anomalies are triggered whp.

Breadth first decoding. It may seem puzzling how decoding could reliably recover from a state where $A \subseteq S$ for some anomaly A. Assume the centre of A stores exactly the keys from A and consider the next time a key $x \in A$ is removed from S. Then A is triggered and it will then appear as though $x = \bigoplus_{x' \in A\setminus\{x\}} x'$ is stored in bucket i. Since i looks pure, x may be detected at i and hence promptly readded to S. This would indeed be a fatal problem if decode would maintain the set of buckets to be processed (i.e. those that look pure) in a LIFO queue. Instead, decode proceeds in rounds and a bucket that attains the $\mathsf{looksPure}$ status is only considered in the next round after all buckets that looked pure at the beginning of the round have been processed. Such a "breadth first" way of considering buckets allows for useful work to be done (including the removal of further keys from A) before the centre bucket i is considered.

2.2 Isolating Anomalies. Let $\mathcal{A}$ be the set of all native anomalies. In the following we prove that only anomalies from $\mathcal{A}$ are triggered during decoding, that those anomalies have canonical properties and do not interact. This will involve several union bound arguments that are similar to each other in structure. As a warm-up we bound $\mathbb{E}[\mathcal{A}]$.

Let us be precise about how a native anomaly arises from the underlying family $(h_j(x))_{x\in[u],j\in[k]}$ of independent random variables. For any $\ell \geq 3$, any set $\{x_1, \dots, x_{\ell-1}\} \subseteq S_0$ and any sequence $j_1, \dots, j_\ell \in [k]$ we call $(\{x_1, \dots, x_{\ell-1}\}, j_1, \dots, j_\ell)$ an *anomaly blueprint*. This blueprint is *realised* if $h_{j_1}(x_1) = \dots = h_{j_\ell}(x_\ell)$ where $x_\ell := x_1 \oplus \dots \oplus x_{\ell-1}$. In that case $A = \{x_1, \dots, x_\ell\}$ is a native anomaly. Conversely, every native anomaly realises at least one blueprint (a native anomaly with no foreign key realises at least ℓ blue prints, corresponding to its subsets of size $\ell - 1$). Thus $|\mathcal{A}|$ is at most the number of realised blueprints. There are $\binom{m}{\ell-1}k^\ell$ blueprints with parameter ℓ and each is realised with probability exactly $n^{-\ell+1}$. Let $\mathcal{P}$ be the set of all anomaly blueprints and let E_P for $P \in \mathcal{P}$ be the event that blueprint P is realised. Recall that in the context we are operating we have $c := \frac{m}{n} < c_k^\triangle - \varepsilon < 1$. We can compute

$$\mathbb{E}[|\mathcal{A}|] \leq \mathbb{E}[\,|\{P \in \mathcal{P} \mid P \text{ is realised}\}|\,] = \sum_{P\in\mathcal{P}} \Pr[E_P] = \sum_{\ell\geq 3} \binom{m}{\ell-1} k^\ell \cdot n^{-\ell+1}$$

$$\leq \sum_{\ell\geq 3} \frac{m^{\ell-1}}{(\ell-1)!} k^\ell n^{-\ell+1} = k \sum_{\ell\geq 3} \frac{(ck)^{\ell-1}}{(\ell-1)!} \leq k \sum_{\ell\geq 0} \frac{(ck)^\ell}{\ell!} = k\mathrm{e}^{ck} = \mathcal{O}(1). \tag{2.1}$$

We now show that whp no anomaly $A \in \mathcal{A}$ is (ii) too large, (iii) contains keys sharing a bucket other than the centre or (iv) intersects other anomalies in $\mathcal{A}$. We use the notation $h(A) := \bigcup_{x\in A} h(x)$.

LEMMA 2.1. *The following holds whp.*

(i) $\forall i \in [n]\colon |\{x \in S_0 \mid i \in h(x)\}| \leq \log n$.

(ii) $\forall A \in \mathcal{A}\colon |A| \leq \log n$.

[4] More precisely: $h(x_j)$ should contain i an odd number of times.

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

(iii) $\forall A \in \mathcal{A}$: $|h(A)| = (k-1)|A| + 1$.

(iv) $\forall A_1 \neq A_2 \in \mathcal{A}$: $h(A_1) \cap h(A_2) = \varnothing$.

Proof. (i) It is well-known that when n balls are randomly thrown into n bins then the expected maximum load of a bin is $\mathcal{O}(\frac{\log n}{\log\log n})$ whp [Gon81, Mit96], which implies that in our setting every bucket stores $\mathcal{O}(\frac{\log n}{\log\log n})$ keys whp. We give a short self-contained proof nonetheless. Let p_{ij} be the probability that a specific bin $i \in [n]$ stores at least $j \in [n]$ keys. A union bound and Stirling's formula gives

$$p_{ij} \leq \binom{km}{j} n^{-j} \leq \frac{(km)^j}{j!} n^{-j} \leq \frac{k^j}{j!} \leq \frac{(k\mathrm{e})^j}{j^j}.$$

For $j = 6\frac{\log n}{\log\log n}$ we get for large n and using $x^{\frac{1}{\log x}} = 2$ that

$$p_{ij} \leq \frac{(k^6\mathrm{e}^6)^{\frac{\log n}{\log\log n}}}{(6\frac{\log n}{\log\log n})^{6\frac{\log n}{\log\log n}}} \leq \frac{2^{\log n}}{(\sqrt{\log n})^{6\frac{\log n}{\log\log n}}} \leq \frac{n}{2^{3\log n}} = n^{-2}.$$

Summing over all i implies that no bin stores $\omega(\frac{\log n}{\log\log n})$ keys whp.

(ii) Since a native anomaly of size ℓ with centre i requires $\ell - 1$ keys from S_0 to be stored in i, the claim follows from (i).

(iii) Let $A \in \mathcal{A}$ be an anomaly and $\ell = |A|$. There are $k\ell$ relevant hash values. The centre of A occurs as a hash value ℓ times, hence there are at most $(k-1)\ell + 1$ distinct hash values. For there to be at most $(k-1)\ell$ distinct hash values, an additional identity of two hash values is needed. Since there are at most $\binom{k\ell}{2}$ potential identities that are realised with probability $\frac{1}{n}$ each, we get with calculations similar to (2.1)

$$\begin{aligned}\Pr[\exists A \in \mathcal{A} : h(\mathcal{A}) \leq (k-1)|A|] &\leq \sum_{\ell\geq 3} \binom{m}{\ell-1} k^\ell n^{-\ell+1} \binom{k\ell}{2} \frac{1}{n} \\ &\leq \sum_{\ell\geq 3} \frac{n^{\ell-1}}{(\ell-1)!} \frac{k^\ell}{n^\ell} k^2\ell^2 \leq \frac{1}{n} \sum_{\ell\geq 3} \frac{k^{\ell+2}\ell^2}{(\ell-1)!} = \mathcal{O}(1/n).\end{aligned}$$

(iv) The main complication stems from the possibility that A_1 and A_2 may share some keys. We distinguish four cases.

Case 1: Shared centres. Consider the event E_1 that there exist $A_1, A_2 \in \mathcal{A}$ with $A_1 \neq A_2$ and the same centre. Assume $|A_1| = \ell_1$, $|A_2| = \ell_2$, $|A_1 \cap A_2| = \bar{\ell}$ and wlog $A_2 \setminus A_1 \neq \varnothing$.
The set A_2 can be uniquely identified by $\bar{\ell}$ keys from A_1 and $\ell_2 - \bar{\ell} - 1$ keys from S_0. We argue similar to Equation (2.1).

$$\begin{aligned}\Pr[E_1] &\leq \sum_{\ell_1\geq 3} \sum_{\ell_2\geq 3} \sum_{0\leq\bar{\ell}\leq\min(\ell_1,\ell_2-1)} \binom{m}{\ell_1-1} \binom{\ell_1}{\bar{\ell}} \binom{m}{\ell_2-\bar{\ell}-1} k^{\ell_1+\ell_2-\bar{\ell}} n^{\ell_1+\ell_2-\bar{\ell}-1} \\ &\leq \sum_{\bar{\ell}\geq 0} \sum_{\ell_1\geq\bar{\ell}} \sum_{\ell_2\geq\bar{\ell}+1} \frac{n^{\ell_1-1}}{(\ell_1-1)!} \frac{\ell_1!}{\bar{\ell}!(\ell_1-\bar{\ell})!} \frac{n^{\ell_2-\bar{\ell}-1}}{(\ell_2-\bar{\ell}-1)!} k^{\ell_1+\ell_2-\bar{\ell}} n^{\ell_1+\ell_2-\bar{\ell}-1} \\ &\leq \frac{1}{n} \sum_{\bar{\ell}\geq 0} \sum_{\ell_1\geq\bar{\ell}} \sum_{\ell_2\geq\bar{\ell}+1} \frac{\ell_1 k^{\ell_1+\ell_2-\bar{\ell}}}{\bar{\ell}!\,(\ell_1-\bar{\ell})!\,(\ell_2-\bar{\ell}-1)!} \\ &\leq \frac{1}{n} \sum_{\bar{\ell}\geq 0} \frac{k^{\bar{\ell}}}{\bar{\ell}!} \sum_{\ell_1\geq\bar{\ell}} \frac{\ell_1 k^{\ell_1-\bar{\ell}}}{(\ell_1-\bar{\ell})!} \sum_{\ell_2\geq\bar{\ell}+1} \frac{k^{\ell_2-\bar{\ell}}}{(\ell_2-\bar{\ell}-1)!} \\ &\leq \frac{k\log n}{n} \Big(\sum_{\ell\geq 0} \frac{k^\ell}{\ell!} \Big)^3 = \mathcal{O}\Big(\frac{\log n}{n}\Big) = \tilde{\mathcal{O}}(1/n)\end{aligned}$$

where we used $\ell_1 \leq \log(n)$ towards the end which we may assume by (ii).

Case 2: $|A_1 \cap A_2| \geq 2$. By (iii) the hashes $h(x)$ and $h(y)$ of two distinct keys x, y in any anomaly A intersect exactly in the centre of A whp. If two anomalies A_1 and A_2 share two keys x and y, they must therefore also share their centre whp. Therefore Case 2 implies Case 1 whp.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Case 3: Distinct centres and $|A_1 \cap A_2| = 1$. Consider the event E_3 that there exist anomalies A_1 and A_2 with distinct centres and one shared key. Let $\ell_1 = |A_1|$ and $\ell_2 = |A_2|$. Now A_2 is uniquely identified by one of the ℓ_1 keys from A_1 and $\ell_2 - 2$ keys from S_0. We get

$$\begin{aligned}\Pr[E_3] &\le \sum_{\ell_1\ge 3}\sum_{\ell_2\ge 3}\binom{m}{\ell_1-1}\ell_1\binom{m}{\ell_2-2}k^{\ell_1+\ell_2}n^{-\ell_1-\ell_2+2}\\ &\le \sum_{\ell_1\ge 3}\sum_{\ell_2\ge 3}\frac{n^{\ell_1-1}}{(\ell_1-1)!}\ell_1\frac{n^{\ell_2-2}}{(\ell_2-2)!}k^{\ell_1+\ell_2}n^{-\ell_1-\ell_2+2}\\ &\le \frac{1}{n}\sum_{\ell_1\ge 3}\frac{k^{\ell_1}\ell_1}{(\ell_1-1)!}\sum_{\ell_2\ge 3}\frac{k^{\ell_2}}{(\ell_2-2)!} \le \mathcal{O}(1/n).\end{aligned}$$

Case 4: Distinct centres and $A_1 \cap A_2 = \varnothing$. Consider the event E_4 that there exist anomalies A_1 and A_2 with distinct centres sharing no key but sharing some $i \in h(A_1) \cap h(A_2)$. Let $\ell_1 = |A_1|$ and $\ell_2 = |A_2|$ and assume wlog that i is not the centre of A_1. One of the $(k-1)\ell_1$ non-centre hashes of keys in A_1 must coincide with one of the $k\ell_2$ hashes from keys in A_2. We get

$$\begin{aligned}\Pr[E_4] &\le \sum_{\ell_1\ge 3}\sum_{\ell_2\ge 3}\binom{m}{\ell_1-1}\binom{m}{\ell_2-1}k^{\ell_1+\ell_2}n^{-\ell_1-\ell_2+2}(k-1)\ell_1 k\ell_2\tfrac{1}{n}\\ &\le \sum_{\ell_1\ge 3}\sum_{\ell_2\ge 3}\frac{n^{\ell_1-1}}{(\ell_1-1)!}\frac{n^{\ell_2-1}}{(\ell_2-1)!}k^{\ell_1+\ell_2}n^{-\ell_1-\ell_2+2}(k-1)\ell_1 k\ell_2\tfrac{1}{n}\\ &\le \frac{1}{n}\sum_{\ell_1\ge 3}\frac{k^{\ell_1+1}\ell_1}{(\ell_1-1)!}\sum_{\ell_2\ge 3}\frac{k^{\ell_2+1}\ell_2}{(\ell_2-1)!} = \mathcal{O}(1/n).\end{aligned}$$

□

We can now derive a concentration bound on the number of native anomalies.

LEMMA 2.2. *There are $|\mathcal{A}| = \mathcal{O}(\log n)$ native anomalies whp.*

Proof. The challenge here is to navigate the fact that anomalies do not occur independently.

Recall the definition of anomaly blueprints. Let $E_{P,i}$ be the event that a blueprint $P \in \mathcal{P}$ is realised at a bucket $i \in [n]$. Importantly, $E_{P,i}$ is simply the event that certain random variables in the family $(h_j(x))_{x\in[u],j\in[k]}$ turn out to be i. If we have a sequence $E_{P_1,i_1},\dots,E_{P_b,i_b}$ of such events pertaining to pairwise distinct buckets $i_1,\dots,i_b \in [n]$ then these events either refer to pairwise distinct random variables and are hence independent, or two events refer to the same random variable and are hence disjoint (i.e. inconsistent). Therefore

$$\Pr\Big[\bigcap_{j\in[b]} E_{P_j,i_j}\Big] \in \Big\{0,\ \prod_{j\in[b]}\Pr[E_{P_j,i_j}]\Big\} \tag{2.2}$$

Now define $E_i := \bigcup_{P\in\mathcal{P}} E_{P,i}$ to be the event that at least one native anomaly has centre i. We can now bound the probability that at least b of these events occur.

$$\begin{aligned}\Pr\Big[\sum_{i\in[n]}\mathbb{1}_{E_i}\ge b\Big] &\le \sum_{I\subseteq[n],|I|=b}\Pr\Big[\bigcap_{i\in I}E_i\Big] \overset{\text{sym}}{=} \binom{n}{b}\Pr\Big[\bigcap_{i=1}^{b}E_i\Big]\\ &= \binom{n}{b}\Pr\Big[\bigcap_{i=1}^{b}\bigcup_{P\in\mathcal{P}}E_{P,i}\Big] = \binom{n}{b}\Pr\Big[\bigcup_{P_1,\dots,P_b\in\mathcal{P}}\bigcap_{i=1}^{b}E_{P_i,i}\Big]\\ &\le \binom{n}{b}\sum_{P_1,\dots,P_b\in\mathcal{P}}\Pr\Big[\bigcap_{i=1}^{b}E_{P_i,i}\Big] \overset{(2.2)}{\le} \binom{n}{b}\sum_{P_1,\dots,P_b\in\mathcal{P}}\prod_{i=1}^{b}\Pr[E_{P_i,i}]\\ &= \binom{n}{b}\prod_{i=1}^{b}\sum_{P\in\mathcal{P}}\Pr[E_{P,i}] = \binom{n}{b}\Big(\sum_{P\in\mathcal{P}}\Pr[E_{P,1}]\Big)^b = \binom{n}{b}\Big(\sum_{\ell\ge 3}\binom{m}{\ell-1}\frac{k^\ell}{n^\ell}\Big)^b\\ &\le \frac{n^b}{b!}\Big(\sum_{\ell\ge 3}\frac{n^{\ell-1}}{(\ell-1)!}\frac{k^\ell}{n^\ell}\Big)^b \le \frac{1}{b!}\Big(\sum_{\ell\ge 3}\frac{k^\ell}{(\ell-1)!}\Big)^b = \frac{(k\mathrm{e}^k)^b}{b!} \le \frac{(k\mathrm{e}^{k+1})^b}{b^b}.\end{aligned}$$

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

For $b = \Omega(\log n)$ the last term is $\mathcal{O}(1/n)$, meaning that only $\mathcal{O}(\log n)$ buckets are the centre of native anomalies whp. Since no two native anomalies share a centre whp by Lemma 2.1 (ii) this implies that there are $\mathcal{O}(\log n)$ native anomalies whp as desired. ∎

LEMMA 2.3. *During decoding, only native anomalies are triggered whp.*

Proof. Assume there is a first time t when a foreign anomaly A_2 is triggered. Let i_2 be its centre and S the set of keys stored in the sketch at time t.

The previously triggered native anomalies may already have introduced some foreign keys to S, but by Lemma 2.1 (iv) these anomalies $A \in \mathcal{A}$ have pairwise disjoint domains $h(A)$, so each bucket stores at most one foreign key. The facts that A_2 contains at least two foreign keys and that all but one of the keys from A_2 must be present in S in order for A_2 to be triggered imply that A_2 contains exactly two foreign keys, one of which is present in S, call it $y_1 \in A_2 \cap S \setminus S_0$, and one of which is absent, call it $y_2 \in A_2 \setminus (S \cup S_0)$. The presence of y_1 in S is due to an anomaly A_1 with some centre i_1 that was triggered previously and must be native by choice of t. We bound the probability for such a situation to exist, distinguishing two cases. For both we define $\ell_1 := |A_1|$ and $\ell_2 := |A_2|$.

Case 1: $i_1 \neq i_2$. We have $A_1 \cap A_2 = \{y_1\}$ because by Lemma 2.1 (iii) no two keys from A_1 can share i_2 as a hash value. The pair (A_1, A_2) is uniquely determined by the $\ell_1 - 1$ native keys from A_1 and the $\ell_2 - 2$ native keys from A_2. The probability for such a pair to exist is

$$\begin{aligned}
&\sum_{\ell_1 \geq 3} \sum_{\ell_2 \geq 3} \binom{m}{\ell_1 - 1} \binom{m}{\ell_2 - 2} k^{\ell_1 + \ell_2} n^{-\ell_1 - \ell_2 + 2} \\
\leq &\sum_{\ell_1 \geq 3} \sum_{\ell_2 \geq 3} \frac{n^{\ell_1 - 1}}{(\ell_1 - 1)!} \frac{n^{\ell_2 - 2}}{(\ell_2 - 2)!} k^{\ell_1 + \ell_2} n^{-\ell_1 - \ell_2 + 2} \leq \frac{1}{n} \sum_{\ell_1 \geq 3} \frac{k^{\ell_1}}{(\ell_1 - 1)!} \sum_{\ell_2 \geq 3} \frac{k^{\ell_2}}{(\ell_2 - 2)!} = \mathcal{O}(1/n).
\end{aligned}$$

Case 2: $i_1 = i_2$. Similar to the proof of Lemma 2.1 (iv) Case 1, there may now be some number $\bar{\ell} := |A_1 \cap A_2 \cap S_0|$ of shared native keys. Otherwise the computation is similar to Case 1.

$$\begin{aligned}
&\sum_{\ell_1 \geq 3} \sum_{\ell_2 \geq 3} \sum_{\bar{\ell} \leq \min\{\ell_1 - 1, \ell_2 - 2\}} \binom{m}{\ell_1 - 1} \binom{\ell_1 - 1}{\bar{\ell}} \binom{m}{\ell_2 - \bar{\ell} - 2} k^{\ell_1 + \ell_2 - \bar{\ell} - 1} n^{-\ell_1 - \ell_2 + \bar{\ell} + 2} \\
\leq &\sum_{\bar{\ell} \geq 0} \sum_{\ell_1 \geq \bar{\ell} + 1} \sum_{\ell_2 \geq \bar{\ell} + 2} \frac{n^{\ell_1 - 1}}{(\ell_1 - 1)!} \frac{(\ell_1 - 1)!}{\bar{\ell}!\,(\ell_1 - \bar{\ell} - 1)!} \frac{n^{\ell_2 - \bar{\ell} - 2}}{(\ell_2 - \bar{\ell} - 2)!} k^{\ell_1 + \ell_2 - \bar{\ell} - 1} n^{-\ell_1 - \ell_2 + \bar{\ell} + 2} \\
\leq &\frac{1}{n} \sum_{\bar{\ell} \geq 0} \frac{k^{\bar{\ell}}}{\bar{\ell}!} \sum_{\ell_1 \geq \bar{\ell} - 1} \frac{k^{\ell_1 - \bar{\ell} - 1}}{(\ell_1 - \bar{\ell} - 1)!} \sum_{\ell_2 \geq \bar{\ell} - 2} \frac{k^{\ell_2 - \bar{\ell}}}{(\ell_2 - \bar{\ell} - 2)!} \leq \mathcal{O}(1/n).
\end{aligned}$$

Taken together, no such situation arises whp.

∎

LEMMA 2.4. *Let $S_{\mathcal{A}} := \bigcup_{A \in \mathcal{A}} A$ be the set of* anomalous keys. *During decoding, we have $S \subseteq S_0 \cup S_{\mathcal{A}}$ at all times whp.*

Proof. This follows from induction. Initially we have $S = S_0$. Any regular decoding step removes an element from S. Any anomalous decoding step adds a key $y \in A$ for some anomaly A that has been triggered. By Lemma 2.3 we have $A \in \mathcal{A}$ and hence $y \in S_{\mathcal{A}}$, maintaining the invariant. ∎

2.3 Working around anomalies. Let $S_{\mathcal{A}} := \bigcup_{A \in \mathcal{A}} A$ be the set of *anomalous keys* and $B_{\mathcal{A}} = h(S_{\mathcal{A}})$ the set of *anomalous buckets*. Consider a variant decode' of decode (see Figure 2) that receives the set $B_{\mathcal{A}}$ of anomalous buckets as a parameter and ignores these buckets, say by pretending that no $i \in B_{\mathcal{A}}$ ever satisfies looksPure(i). Similar to S, we use S' to track the set of keys stored in the sketch over time when decode' is used. No anomalous decoding steps can occur in decode', because the first anomalous decoding step would have to be at the centre of a native anomaly, but these centres are contained in $B_{\mathcal{A}}$ and banned from consideration. In particular, elements are only ever removed from S', never added.

Recall that by a *round* of decode we mean one iteration of the while-loop. Rounds typically comprise many decoding steps. To ensure the rth round is well-defined for each $r \in \mathbb{N}$ we imagine that if and when the algorithm terminates (because both Q and Q_{next} are empty) an infinite number of further rounds take place that contain no steps.

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

We now show that decode correctly identifies at least as many keys from S_0 as decode'. For this we define S_r for $r \in \mathbb{N}_0$ to be the set of keys stored in the sketch at the start of round $r+1$ when decode is used and S'_r to be the corresponding set when decode' is used.

LEMMA 2.5. *We have $S_r \cap S_0 \subseteq S'_r$ for all $r \in \mathbb{N}_0$ whp.*

Proof. We proceed by induction. At the start of round 1 we have $S_0 = S'_0$ so there is nothing to show. Now assume that at the start of some round r we have $S_r \cap S_0 \subseteq S'_r$. To show $S_{r+1} \cap S_0 \subseteq S'_{r+1}$ we consider different cases for $x \in S_{r+1} \cap S_0$. We may assume that the high-probability guarantees from Lemmas 2.3 and 2.4 hold.

Case 1: $x \in A$ for some $A \in \mathcal{A}$. We have $h(x) \subseteq B_{\mathcal{A}}$, i.e. the buckets of x are banned from consideration in decode'. Since $x \in S_0 = S'_0$ and x can never be toggled in decode' we have $x \in S'_{r+1}$ as well.

Case 2: x is not part of a native anomaly. Combined with Lemma 2.3, x is not contained in any anomaly that is triggered and could not have been readded to S. It was therefore already in S when round r started, meaning $x \in S_r \cap S_0$. The induction hypothesis gives $x \in S'_r$. We have to show $x \in S'_{r+1}$. Assume for contradiction that $x \notin S'_{r+1}$. Then x was removed in round r of decode'. Thus one of its buckets $b \in h(x)$ was in Q at the start of round r of decode'. Hence $b \notin B_{\mathcal{A}}$ and we had looksPure(b) at some prior time. No anomaly can be triggered at b by Lemma 2.3 and looksPure(b) really means that only one key is stored in b. Hence x is the only key in S'_r with b as a hash. By induction hypothesis, x is the only key in $S_r \cap S_0$ with b as a hash. Moreover, since $S_r \setminus S_0 \subseteq S_{\mathcal{A}}$ by Lemma 2.4 we have $h(S_r \setminus S_0) \subseteq B_{\mathcal{A}} \not\ni b$ so x is the only key in S_r with b as a hash and looksPure(b) holds at the start of round $r+1$ of decode. This implies that b is in Q at the start of round r of decode. Again using that no triggered anomaly can add keys to bucket b, we conclude that x is detected and removed during round $r+1$ of decode. Moreoever, x cannot be readded afterwards since $x \notin S_{\mathcal{A}}$. This implies $x \notin S_{r+1}$, contradicting the choice of $x \in S_{r+1} \cap S_0$. Since the assumption $x \notin S'_{r+1}$ led to this contradiction we have $x \in S'_{r+1}$ as desired.

□

On the other hand we will show in Section 2.4 that decode' succeeds in decoding everything except keys in $S_{\mathcal{A}}$ and in fact does so in a polylogarithmic number of rounds:

LEMMA 2.6. *With high probability, decode' achieves $S'_R \subseteq S_{\mathcal{A}}$ for some $R = \tilde{\mathcal{O}}(1)$.*

Before showing how this implies our main theorem we deal with a technicality concerning the implementation of Q and Q_{next}. As the names suggest, we have queues in mind, such as a LIFO or FIFO queue. However, since we cannot afford to check if an element is already in Q_{next} whenever we are about to add something to Q_{next} this effectively implements Q and Q_{next} as multisets. A duplicated bucket i in Q_{next} means a duplicated execution of the for-loop for bucket i in the next round. None of the previous arguments hinge on this, but one might worry that with excessive duplication the running time gets out of hand. We are reluctant to resolve the issue by using a set data structure for Q_{next}, because this would compromise the simplicity of decode. Moreover the issue can be resolved with the following simple Lemma.

LEMMA 2.7. *Assume an implementation of decode realises Q and Q_{next} as multisets, e.g. using FIFO queues. Then whp the following is true for all $A \in \mathcal{A}$. Together Q and Q_{next} never contain more than two copies of the centre i of A. If they contain two copies of i, then i stores at most one key.*

Proof. Since no anomaly other than A affects i by Lemma 2.1 (iv) and Lemma 2.3, there are only two reasons for adding i to Q_{next}:

(i) The anomaly A is triggered, meaning the state of i changed such that i now stores $|A|-1$ keys from A and no other key, or

(ii) i stores only a single key.

Reason (i) may occur several times but the necessary state change in between two occurrences must include the addition of a key and the removal of a key. A key can only be added to i due to an anomalous decoding step at i, which consumes a copy of i from Q, maintaining the invariant. While reason (ii) may push a second copy of i into Q_{next}, this can only happen once since no anomalous decoding steps can add keys to i afterwards (recall that $|A| \geq 3$). □

Proof. [Proof of Theorem 1.1] We may assume that the high probability events from all previous lemmas hold. Let $R = \tilde{\mathcal{O}}(1)$ be the number of rounds from Lemma 2.6 needed until $S'_R \subseteq S_{\mathcal{A}}$. Since $S_R \cap S_0 \subseteq S'_R$ by Lemma 2.5 and $S_R \setminus S_0 \subseteq S_{\mathcal{A}}$ by Lemma 2.4 we have $S_R \subseteq S_{\mathcal{A}}$ as well, i.e. only anomalous keys might remain after R rounds of decode. We now show that two more rounds suffice (i.e. $S_{R+2} = \varnothing$) by showing $S_{R+2} \cap A = \varnothing$ for any $A \in \mathcal{A}$.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Since native anomalies have non-overlapping domain $h(A)$ we may consider each A in isolation. Let i be the centre of A. Consider the beginning of round $R+1$ (when $Q_{\text{next}} = \varnothing$). If Q contains two copies of i, then by Lemma 2.7 we have $|S_R \cap A| = 1$ and this single key is clearly removed in the next round. Otherwise Lemma 2.7 guarantees that there is at most one copy of i in Q. Each $x \in A$ is the only key stored in the $k-1$ buckets $h(x) \setminus \{i\}$ by Lemma 2.1 (iii). These buckets "lookPure" and are hence all contained in Q. Therefore, every $x \in A$ is removed within round $R+1$ (at least once). When bucket i is processed, at most one key from A is added. This leaves us with at most one key from A after $R+1$ rounds and hence no key from A after $R+2$ rounds. This concludes the proof that $S = \varnothing$ after $\tilde{\mathcal{O}}(1)$ rounds of decode and hence that decode terminates with $S_{\text{dec}} = S_0$ whp.

A final issue is the running time. We assume Q and Q_{next} are implemented as queues. Let a be the total number of anomalous decoding steps. By Lemma 2.2 there are $\tilde{\mathcal{O}}(1)$ native anomalies whp, by Lemma 2.3 no other anomalies are ever triggered whp and by Lemma 2.7 each anomaly can lead to at most one anomalous decoding step per round whp. Hence $a = \tilde{\mathcal{O}}(1)$ whp. Since regular decoding steps remove a key and anomalous steps add a key, there are $m + a$ regular steps, giving $m + 2a$ decoding steps in total whp. The total number of entries added to Q and Q_{next} is then at most $n + (k-1)(m+2a) = \mathcal{O}(n)$, which accounts for n additions before the while-loop and $k-1$ additions per decoding step. Since every iteration of the for-loop consumes an element from Q, there are $\mathcal{O}(n)$ for-loop iterations whp. ∎

2.4 Analysis of decode'. The arguments used in the following proof of Lemma 2.6 should not be regarded as completely novel. The fact that most keys can be removed is closely related to the analysis of cores in hypergraphs as discussed by Molloy [Mol05] and the required number of rounds of peeling has been studied in a similar case in more detail by Jiang, Mitzenmacher and Thaler [JMT16] who prove that $\Theta(\log \log n)$ rounds are necessary and sufficient. We adapt these existing works to our setting with anomalies.

We recall some facts about hypergraph peeling closely related to our setting. The hypergraph to consider here is $H = ([n], \{h(x) \mid x \in S_0\})$. To avoid parallel terminologies, we continue to call $i \in [n]$ a bucket (rather than a vertex) and speak of a key x (effectively referring to the hyperedge $h(x)$). We do however adopt graph theoretic notions such as the *incidence* of a bucket i to a key x (meaning $i \in h(x)$), the degree of a bucket (its number of incidences) or the r-neighbourhood of a bucket or key (the set of buckets and keys reachable by traversing at most r hyperedges).

The peeling process on H proceeds in rounds. In every round the set of buckets $B_1 \subseteq [n]$ of degree 1 is determined. Then all keys incident to a bucket from B_1 are removed. This may cause further buckets to lose incidences, creating new buckets of degree 1, which are then handled in the next round. If this process eventually removes all keys, then the original H is called *peelable*. Peelability is for instance exploited to decode IBLTs [GM11], construct error correcting codes [LMSS01] and to solve random linear systems to construct perfect hash functions [BPZ13] or Bloom filter alternatives [GL20].

A density threshold $c_k^{\triangle}$ for peelability is known, meaning a fully random k-uniform hypergraph (like H above) is peelable whp if $\frac{m}{n} < c_k^{\triangle} - \varepsilon$ and not peelable whp if $\frac{m}{n} > c_k^{\triangle} + \varepsilon$ [Mol05]. Moreover, the following lemma guarantees that below the threshold a constant number of rounds suffice to remove most keys whp:

LEMMA 2.8. (MOLLOY [MOL05, LEMMA 3]) *For any $\varepsilon, \delta > 0$, there exists $R \in \mathbb{N}$ such that after peeling a k-uniform hypergraph with hyperedge density $\frac{m}{n} < c_k^{\triangle} - \varepsilon$ for R rounds at most δn hyperedges remain whp.*[5]

Our algorithm decode' behaves almost exactly like a peeling algorithm. The only substantial difference is that the buckets from $B_{\mathcal{A}}$ are never considered regardless of their degree. As it turns out, this disturbance is too weak to affect the guarantee given in Lemma 2.8, as we show now.

LEMMA 2.9. *For any $\delta > 0$, there exists $R \in \mathbb{N}$ such that $|h(S'_R)| \leq \delta n$ whp.*

Proof. The density condition $\frac{m}{n} < c_k^{\triangle} - \varepsilon$ is part of the requirement of Theorem 1.1 – the context in which we are operating. Without the complication of anomalous buckets we could obtain a constant R such that $|S'_R| \leq \frac{\delta}{2k} n$ whp by Lemma 2.8. Since peeling is a local algorithm, a key x is only affected by the restriction regarding $B_{\mathcal{A}}$ if there is some $i \in B_{\mathcal{A}}$ within the R-neighbourhood of x. Since the maximum degree of any bucket is at most $\log n$ whp by Lemma 2.1 and because $|B_{\mathcal{A}}| \leq k|S_{\mathcal{A}}| \leq k \log(n)|\mathcal{A}| = \mathcal{O}(\log^2(n))$ by Lemma 2.2 there are whp at most $\mathcal{O}(\log^{2+R}(n))$ buckets in the R-neighbourhoods of buckets in $B_{\mathcal{A}}$ that could be affected in this way. We obtain $|S'_R| \leq \frac{\delta}{2k} n + \mathcal{O}(\log^{2+R}(n)) \leq \frac{\delta}{k} n$ whp and therefore $|h(S'_R)| \leq k|S'_R| \leq \delta n$ whp as desired. ∎

For a set $I \subseteq [n]$ of buckets let $S_I := \{x \in S_0 \mid h(x) \subseteq I\}$ be the set of keys *induced* by I.

[5] Strictly speaking, Molloy's Lemma only claims a probability of $1 - o(1)$. However, the tool utilised in his proof (Azuma's inequality in his Lemma 7) is strong enough to support our "*whp*".

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

LEMMA 2.10. *There exist constants $N \in \mathbb{N}$ and $\delta, \gamma > 0$ such that, whp, any set $I \subseteq [n]$ of buckets with $N \leq |I| \leq \delta n$ satisfies $|S_I| \leq (2-\gamma)|I|/k$.*

Proof. We start by bounding the probability p_i that there exists a set I of size $i := |I|$ with $|S_I| \geq s$ where $s := s(i) := \lceil (2-\gamma)i/k \rceil$, using a union bound. At the line break we use that $i = \Theta(s)$ and that hence $\mathrm{e}^{i+s}(\frac{s}{i})^s \leq C^i$ for a suitable constant C. We also use $k \geq 3$ and choose $\gamma = \frac{1}{4}$.

$$\begin{aligned} p_i &\leq \binom{n}{i}\binom{m}{s}\Big(\frac{i}{n}\Big)^{ks} \leq \Big(\frac{ne}{i}\Big)^i\Big(\frac{ne}{s}\Big)^s\Big(\frac{i}{n}\Big)^{ks} = \mathrm{e}^{i+s}\Big(\frac{i}{s}\Big)^s \cdot \Big(\frac{n}{i}\Big)^i\Big(\frac{n}{i}\Big)^s\Big(\frac{i}{n}\Big)^{ks} \\ &\leq C^i\Big(\frac{n}{i}\Big)^i\Big(\frac{i}{n}\Big)^{(k-1)(2-\gamma)i/k} \leq C^i\Big(\frac{i}{n}\Big)^{-i+\frac{2}{3}(2-\gamma)i} \leq C^i\Big(\frac{i}{n}\Big)^{(\frac{1}{3}-\frac{2}{3}\gamma)i} \leq C^i\Big(\frac{i}{n}\Big)^{i/6} = \Big(\frac{iC^6}{n}\Big)^{i/6}. \end{aligned}$$

We fix $N := 18$ and $\delta := C^{-6}/2$. This allows for the following union bound on the probability that a set I of some size $N \leq |I| \leq \delta n$ induces s or more keys.

$$\begin{aligned} \sum_{i=N}^{\delta n} p_i &\leq \sum_{i=N}^{\delta n}\Big(\frac{iC^6}{n}\Big)^{i/6} = \sum_{i=N}^{\sqrt{n}}\Big(\frac{iC^6}{n}\Big)^{i/6} + \sum_{i=\sqrt{n}+1}^{\delta n}\Big(\frac{iC^6}{n}\Big)^{i/6} \\ &\leq \sum_{i=N}^{\sqrt{n}}\Big(\frac{\sqrt{n}C^6}{n}\Big)^{N/6} + \sum_{i=\sqrt{n}+1}^{\delta n}\Big(\frac{\delta nC^6}{n}\Big)^{\sqrt{n}/6} \leq \sqrt{n}\cdot\Big(\frac{C^6}{\sqrt{n}}\Big)^3 + n\cdot\Big(\frac{1}{2}\Big)^{\sqrt{n}/6} = \mathcal{O}(1/n). \end{aligned} \tag{2.3}$$

□

LEMMA 2.11. *There exists a constant $\delta > 0$ such that the following holds whp. For any set of keys $S^* \subseteq S_0 \setminus S_{\mathcal{A}}$ and $I := h(S^*)$, $i := |I|$, $i \leq \delta n$, $s := |S^*|$ we have $s < (2i - |I \cap B_{\mathcal{A}}|)/k$.*

Proof. We use a union bound from the perspective of the set I. Concretely we bound for fixed $i, a \in \mathbb{N}$ the probability $p_{i,a}$ that there exists a set $I \subseteq [n]$ of size i that satisfies $|I \cap B_{\mathcal{A}}| \geq a$ as well as $|S_I \setminus S_{\mathcal{A}}| \geq s$ for $s := s(i,a) := \max(\frac{i}{k}, \lceil (2i-a)/k \rceil)$, i.e. I contains at least a anomalous buckets and induces at least s keys from $S_0 \setminus S_{\mathcal{A}}$. It then suffices to show that the sum over all $p_{i,a}$ is $\mathcal{O}(1/n)$. Note that we may assume $s \geq \frac{i}{k}$ because of "$I = h(S^*)$". This ensures $s = \Theta(i)$.

We enumerate all ways in which I might arise (though including some inconsistent combinations in our counting), thereby explaining Equation (2.4) below, from left to right.

- There are $\binom{n}{i}$ ways to select I,
- and $\binom{m}{s}$ ways to select a set of s keys from $S_0 \setminus S_{\mathcal{A}}$ to be induced by I.
- We specify a number $d \leq a$ of disjoint native anomalies that are to contribute to $I \cap B_{\mathcal{A}}$ (note that by Lemma 2.1 (iv) we need not worry about the possibility of intersecting native anomalies).
- For each $j \in [d]$ we specify properties of the jth anomaly A_j, namely:
  - A number $a_j \geq 1$ of elements from $I \cap B_{\mathcal{A}}$ that A_j accounts for. These numbers should satisfy $\sum_j a_j = a$. Each a_j is a lower bound on $|I \cap h(A_j)|$.
  - The size $\ell_j := |A_j|$ of A_j.
  - A set of $\ell_j - 1$ keys from S_0 that uniquely determine A_j itself.
  - For each $x \in A_j$, which of its k hashes should point to the anomaly's centre.
  - And finally, which subset of the k hashes of $x \in A_j$ must fall within I.

The probability for such a precisely specified constellation is either 0 (if the specification is inconsistent) or has a simple form shown in the following formula. The term $(\frac{i}{n})^{ks}$ accounts for the selected keys being induced by I, $n^{-\ell_j+1}$ accounts for the keys in A_j forming an anomaly and $(\frac{i}{n})^{a_j}$ accounts for the selected hashes from keys in A_j actually falling into I. It should be clear that these three probabilities are independent. We obtain the following.

$$\begin{aligned} p_{i,a} &\leq \underbrace{\tbinom{n}{i}\tbinom{m}{s}\sum_{d\leq a}\sum_{a_1+\ldots+a_d=a}\sum_{\ell_1\geq 3}\tbinom{m}{\ell_1-1}(k2^k)^{\ell_1}\cdots\sum_{\ell_d\geq 3}\tbinom{m}{\ell_d-1}(k2^k)^{\ell_d}}_{\text{sum over all events contributing to the union bound}} \cdot \underbrace{(\tfrac{i}{n})^{ks}\prod_{j\in[d]} n^{-\ell_j+1}(\tfrac{i}{n})^{a_j}}_{\text{probability of the event}} \\ &\leq \tbinom{n}{i}\tbinom{m}{s}(\tfrac{i}{n})^{ks}\sum_{d\leq a}\sum_{a_1+\ldots+a_d=a}\underbrace{\prod_{j\in[d]}\sum_{\ell_j\geq 3}\tbinom{m}{\ell_j-1}(k2^k)^{\ell_j} n^{-\ell_j+1}(\tfrac{i}{n})^{a_j}}_{(*)} \end{aligned} \tag{2.4}$$

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Let us continue to simplify $(*)$, using the constant $C := k2^k e^{k2^k}$.

$$(*) \le \sum_{\ell_j \ge 3} \frac{n^{\ell_j - 1}}{(\ell_j - 1)!} (k2^k)^{\ell_j} n^{-\ell_j + 1} (\tfrac{i}{n})^{a_j} \le \Big(\frac{i}{n}\Big)^{a_j} \sum_{\ell_j \ge 3} \frac{(k2^k)^{\ell_j}}{(\ell_j - 1)!} \le C\Big(\frac{i}{n}\Big)^{a_j}.$$

We continue the computation in Equation (2.4) using that any $a \ge 1$ can be written as the sum of a sequence of positive integers in precisely 2^{a-1} ways, and $a = 0$ can be written in precisely one way as the empty sum.

$$\begin{aligned} p_{i,a} &\le \tbinom{n}{i}\tbinom{m}{s}(\tfrac{i}{n})^{ks} \sum_{d \le a} \sum_{a_1 + \ldots + a_d = a} \prod_{j \in [d]} C\Big(\frac{i}{n}\Big)^{a_j} \le \tbinom{n}{i}\tbinom{m}{s}(\tfrac{i}{n})^{ks} \sum_{d \le a} \sum_{a_1 + \ldots + a_d = a} C^a \Big(\frac{i}{n}\Big)^a \\ &\le \tbinom{n}{i}\tbinom{m}{s}(\tfrac{i}{n})^{ks} 2^a C^a \Big(\frac{i}{n}\Big)^a \le \Big(\frac{en}{i}\Big)^i \Big(\frac{en}{s}\Big)^s \Big(\frac{i}{n}\Big)^{ks} 2^a C^a \Big(\frac{i}{n}\Big)^a \\ &\le e^{i+s} \Big(\frac{i}{s}\Big)^s 2^a C^a \cdot \Big(\frac{n}{i}\Big)^i \Big(\frac{n}{i}\Big)^s \Big(\frac{i}{n}\Big)^{ks} \Big(\frac{i}{n}\Big)^a \le (C')^i \cdot \Big(\frac{i}{n}\Big)^{(k-1)s + a - i} \end{aligned}$$

where C' is another constant (here we use $a \le i$ and $s = \Theta(i)$). We bound the exponent using $s \ge (2i - a)/k$ (and hence $ks + a \ge 2i$) as well as $k \ge 3$ as follows.

$$(k-1)s + a - i = \tfrac{k-1}{k}(ks + a) + a/k - i \ge \tfrac{k-1}{k} \cdot 2i + a/k - i = \tfrac{k-2}{k} i + a/k \ge i/3 + a/k.$$

Considering that the exponent was an integer we finally obtain

$$p_{i,a} \le (C')^i \cdot \Big(\frac{i}{n}\Big)^{\lceil i/3 + a/k \rceil} =: q_{i,a}.$$

To conclude the prove we need to bound $P := \sum_{1 \le i \le \delta n} \sum_{a \ge 0} p_{i,a}$ for some $\delta > 0$ of our choosing. The sum over a is effectively a geometric sum with

$$P \le \sum_{1 \le i \le \delta n} \sum_{a \ge 0} q_{i,a} \le \sum_{1 \le i \le \delta n} k \cdot q_{i,0} \sum_{a \ge 0} \Big(\frac{i}{n}\Big)^a \le k \cdot \sum_{1 \le i \le \delta n} q_{i,0} \cdot \frac{1}{1 - i/n} \le \frac{k}{1 - \delta} \cdot \sum_{1 \le i \le \delta n} q_{i,0}.$$

To bound the sum over the $q_{i,0}$ we have to make use of the "$\lceil \cdot \rceil$" for the leading terms, e.g. like this:

$$\sum_{1 \le i \le \delta n} q_{i,0} \le \mathcal{O}(1/n) + \sum_{9 \le i \le \delta n} \Big(\frac{i(C')^3}{n}\Big)^{i/3}.$$

To bound the remaining sum the same idea as in Equation (2.3) works. ∎

Proof. [Proof of Lemma 2.6.] We assume that the high probability guarantees from Lemmas 2.9 to 2.11 hold. Let N and γ be the constants from Lemma 2.10. Moreover, Lemmas 2.10 and 2.11 each guarantee the existence of a constant named "δ". Let δ be the smaller of the two and apply Lemma 2.9 for this δ. This yields another constant $R = R(\delta)$ such that $|h(S'_R)| \le \delta n$. We refer to the first R rounds as the *early rounds* of peeling.

Now consider any round $r \ge R$ such that $S'_r \setminus S_{\mathcal{A}} \ne \varnothing$. We define $I := h(S'_r \setminus S_{\mathcal{A}})$ and $i := |I|$. Since $S'_r \subseteq S'_R$ we have $i \le |h(S'_R)| \le \delta n$ so we may apply both Lemmas 2.10 and 2.11 to I (see below). It is useful to distinguish three kinds of buckets in I.

- $I_{\mathcal{A}} := I \cap B_{\mathcal{A}}$: anomalous buckets in I.
- $I_1 := \{b \in I \mid \exists!(x, j) \in S_I \times [k] \colon h_j(x) = b\}$ where "$\exists!$" means "there exists exactly one". These are buckets not in $I_{\mathcal{A}}$ that have degree 1 with respect to S_I.
- $I_{2+} := I \setminus (I_{\mathcal{A}} \cup I_1)$: Buckets not in $I_{\mathcal{A}}$ that have degree 2 or more with respect to S_I.

Denote the numbers of these buckets with $i_{\mathcal{A}}, i_1, i_{2+}$, respectively. We have

(i) $i = i_1 + i_{2+} + i_{\mathcal{A}}$ by definition.

(ii) $k|S_I| \ge k|S_I \setminus S_{\mathcal{A}}| \ge i + i_{2+}$ since each $b \in I$ is hashed to at least once and each $b \in I_{2+}$ is hashed to at least twice, both by keys from $S_I \setminus S_{\mathcal{A}}$.

(iii) $|S_I| \le (2 - \gamma)i/k$ and equivalently $2i \ge \frac{2}{2-\gamma} k|S_I|$ by Lemma 2.10 if $i \ge N$.

(iv) $|S_I \setminus S_{\mathcal{A}}| < (2i - i_{\mathcal{A}})/k$ by Lemma 2.11.

We distinguish two further types of rounds depending on i. We show $i_1 > 0$ in both cases.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited

Intermediate rounds with $i = \omega(\log^2 n)$. We compute

$$i_1 \overset{\text{(i)}}{=} i - i_{2+} - i_{\mathcal{A}} = 2i - i - i_{2+} - i_{\mathcal{A}} \overset{\text{(iii)}}{\geq} \tfrac{2}{2-\gamma} k|S_I| - i - i_{2+} - i_{\mathcal{A}}$$
$$\overset{\text{(ii)}}{\geq} \tfrac{2}{2-\gamma}(i + i_{2+}) - (i + i_{2+}) - i_{\mathcal{A}} \geq \tfrac{\gamma}{2-\gamma}(i + i_{2+}) - |B_{\mathcal{A}}| \geq \tfrac{\gamma}{2-\gamma} i - \mathcal{O}(\log^2 n) = \Omega(i).$$

In the end we used the case assumption and a bound of $\mathcal{O}(\log n)$ on both the size and the number of native anomalies that hold whp by Lemma 2.1 (ii) and Lemma 2.2.

Late rounds with $i = \tilde{\mathcal{O}}(1)$. We proceed similarly:

$$i_1 \overset{\text{(i)}}{=} i - i_{2+} - i_{\mathcal{A}} = (2i - i_{\mathcal{A}}) - i - i_{2+} \overset{\text{(iv)}}{>} k|S_I \setminus S_{\mathcal{A}}| - i - i_{2+} \overset{\text{(ii)}}{\geq} (i + i_{2+}) - (i + i_{2+}) = 0.$$

The fact that $i_1 > 0$ holds in both cases guarantees a bucket $b \notin I_{\mathcal{A}}$ storing one key. In the next round of decode' at least this bucket will be cleared of its key. Progress only stops when $S'_r \setminus S_{\mathcal{A}} = \varnothing$, i.e. when our goal $S'_r \subseteq S_{\mathcal{A}}$ is reached. Concerning the number of rounds we have

- $R = \mathcal{O}(1)$ *early* rounds.
- $\mathcal{O}(\log n)$ intermediate rounds since $i_1 = \Omega(i)$ actually guarantees that a constant fraction of the buckets in I are cleared by the round.
- $\tilde{\mathcal{O}}(1)$ late rounds since each clears at least one of the $\tilde{O}(1)$ remaining buckets from I.

The sum is $\tilde{\mathcal{O}}(1)$ rounds as claimed. ∎

References

[BPZ13] Fabiano Cupertino Botelho, Rasmus Pagh, and Nivio Ziviani. Practical perfect hashing in nearly optimal space. *Inf. Syst.*, 38(1):108–131, 2013.

[CJ19] Graham Cormode and Hossein Jowhari. L_p samplers and their applications: A survey. *ACM Comput. Surv.*, 52(1):16:1–16:31, 2019.

[EG11] David Eppstein and Michael T. Goodrich. Straggler identification in round-trip data streams via newton's identities and invertible bloom filters. *IEEE Trans. Knowl. Data Eng.*, 23(2):297–306, 2011.

[EGUV11] David Eppstein, Michael T. Goodrich, Frank Uyeda, and George Varghese. What's the difference?: efficient set reconciliation without prior context. In *Proc. SIGCOMM '11*, pages 218–229, 2011.

[Gan07] Sumit Ganguly. Counting distinct items over update streams. *Theor. Comput. Sci.*, 378(3):211–222, 2007.

[GL20] Thomas Mueller Graf and Daniel Lemire. Xor filters: Faster and smaller than Bloom and cuckoo filters. *ACM J. Exp. Algorithmics*, 25:1–16, 2020.

[GM08] Sumit Ganguly and Anirban Majumder. Deterministic k-set structure. *Inf. Process. Lett.*, 109(1):27–31, 2008.

[GM11] Michael T. Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *Proc. 49st Allerton*, pages 792–799, 2011.

[Gon81] Gaston H. Gonnet. Expected length of the longest probe sequence in hash code searching. *Journal of the ACM*, 28(2):289–304, apr 1981.

[JMT16] Jiayang Jiang, Michael Mitzenmacher, and Justin Thaler. Parallel peeling algorithms. *ACM Trans. Parallel Comput.*, 3(1):7:1–7:27, 2016.

[LMSS01] Michael Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A. Spielman. Efficient erasure correcting codes. *IEEE Trans. Inf. Theory*, 47(2):569–584, 2001.

[McG14] Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014.

[Mit96] Michael David Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing.* PhD thesis, Harvard University, 1996.

[Mol05] Michael Molloy. Cores in random hypergraphs and Boolean formulas. *Random Struct. Algorithms*, 27(1):124–135, 2005.

[MTZ03] Yaron Minsky, Ari Trachtenberg, and Richard Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213–2218, 2003.

[Woo14] David P. Woodruff. Data streams and applications in computer science. *Bull. EATCS*, 114, 2014.

Downloaded 04/27/23 to 151.48.223.142 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited