Beyond the Worst-Case Analysis of Algorithms

Edited by
Tim Roughgarden

Contents

1

When Simple Hash Functions Suffice

Kai-Min Chung; Michael Mitzenmacher; Salil Vadhan

Abstract

In this chapter, we describe a semi-random data model under which simple, explicit families of hash functions, such as those that are 2-universal or $O(1)$-wise independent, perform in a way that is nearly indistinguishable from idealized random hashing, where each data item is mapped independently and uniformly to the range. Specifically, we show that it suffices for the data to come from a "block source," whereby each new data item has some "entropy" given the previous ones. This provides a possible explanation for the observation that simple hash functions, including 2-universal hash functions, often perform as predicted by analysis for the idealized model of truly random hash functions, despite generally having noticeably weaker worst-case guarantees.

1.1 Introduction

Hashing is at the core of many fundamental algorithms and data structures, including all varieties of hash tables, Bloom filters and their many variants, summary algorithms for data streams, and many others. Traditionally, applications of hashing are analyzed as if the hash function is a truly random function (a.k.a. "random oracle") mapping each data item independently and uniformly to the range of the hash function. However, this idealized model is arguably unrealistic, because a truly random function mapping $\{0,1\}^n$ to $\{0,1\}^m$ requires an exponential (in n) number of bits to describe.

For this reason, a long line of theoretical work has sought to provide rigorous bounds on performance when explicit families of hash functions are used, e.g. families whose description and computational complexity are polynomial in n and m. The first examples used *2-universal hash families*, which have the property that for every two distinct inputs $x \neq x' \in \{0,1\}^n$, if we choose a random hash function H from the family, the probability that x and x' collide under H (i.e. $H(x) = H(x')$) is at most $1/2^m$. There are 2-universal families where each hash function has a

description length that is linear (in n) and can be evaluated in nearly linear time, and the 2-universal property can be shown to suffice for a number of applications of hashing. A stronger property sometimes used is s-wise independence, where for every s distinct inputs $x_1, \ldots, x_s \in \{0,1\}^n$, the hash values $H(x_1), \ldots, H(x_s)$ are uniform and independent in $\{0,1\}^m$. However, achieving s-wise independence would require the description length and the evaluation time to be at least linear in $s \cdot m$.

While many beautiful results of this type have been obtained, they are not always as strong as we would like. In some cases, the types of hash functions analyzed can be implemented very efficiently (e.g. universal or $O(1)$-wise independent hash functions), but the performance guarantees are noticeably weaker than for ideal hashing. In other cases, the performance guarantees are (essentially) optimal, but the hash functions are more complex and expensive (e.g. with a super-linear time or space requirement). For example, if at most T items are going to be hashed, then a T-wise independent hash function will have precisely the same behavior as an ideal hash function. But a T-wise independent hash function mapping to $\{0,1\}^m$ requires at least $T \cdot m$ bits to represent, which is often too large. For some applications, it has been shown that less independence, such as $O(\log T)$-wise independence, suffices, but such functions are still substantially less efficient than 2-universal hash functions.

In practice, however, the performance of standard universal hashing often seems to match what is predicted for ideal hashing. Thus, it may not always be necessary to use the more complex hash functions for which this kind of performance can be proven. As in many other examples in this book, this gap between theory and practice may be due to worst-case analysis. Indeed, in some cases, it can be proven that there exist sequences of data items for which universal hashing does not provide optimal performance. But these bad sequences may be pathological cases that are unlikely to arise in practice. That is, the strong performance of universal hash functions in practice may arise from a *combination* of the randomness of the hash function and the randomness of the data.

Of course, doing an average-case analysis, whereby each data item is independently and uniformly distributed in $\{0,1\}^n$, is also very unrealistic (not to mention that it trivializes many applications). In this chapter, we describe an intermediate model, previously studied in the literature on "randomness extractors," that may be an appropriate data model for some hashing applications. Under the assumption that the data fits this model, we will see that relatively weak hash functions achieve essentially the same performance as ideal hash functions.

1.1.1 The Model

We will model the data as coming from a random source in which the data items can be far from uniform and have arbitrary correlations, provided that each (new) data item is sufficiently unpredictable given the previous items. This is formalized by the

notion of a *block source*, where we require that the i-th item (block) X_i has at least some k bits of "entropy" conditioned on the previous items (blocks) $X_1, \ldots, X_{i-1}$. There are various choices for the entropy measure that can be used here; *min-entropy* is used most commonly in the literature on randomness extractors, but most of the results presented here hold even for the less stringent measure of *Rényi entropy*. (See Section 1.2.3 for the formal definitions.)

Block sources seem to be a plausible model for many real-life data sources where we believe that there are is some intrinsic randomness in each data item, provided the entropy k required per-block is not too large. However, in some settings, the data may have structure that violates the block-source property, in which case the results of this chapter will not apply. See Section 1.1.4 for further discussion about the model.

1.1.2 The Results

Here we give a high-level overview of the results presented in this chapter; see Sections 1.2 and 1.3 for the formal treatment of the definitions and results.

It turns out that standard results in the literature on "randomness extractors"[1] already imply that universal hashing performs nearly as well ideal hashing, provided the data items have enough entropy. Specifically, if we have T data items coming from a block source $(X_1, \ldots, X_T)$ where each data item has (Rényi) entropy at least $m + 2\log(T/\epsilon)$ (all logs are base 2 in this chapter) and H is a random 2-universal hash function mapping to $\{0,1\}^m$, then $(H(X_1), \ldots, H(X_T))$ has statistical distance at most ϵ from T uniform and independent elements of $\{0,1\}^m$. Thus, any event that would occur with some probability p under ideal hashing now occurs with probability $p \pm \epsilon$. This allows us to automatically translate existing results for ideal hashing into results for universal hashing in the block-source model.

In many hashing applications, it is possible to improve on the above analysis and reduce the amount of entropy required from the data items. Assuming our hash function has a description size $o(mT)$, then we must have at least $(1 - o(1))m$ bits of entropy per item for the hashing to "behave like" ideal hashing (because the entropy of $(H(X_1), \ldots, H(X_T))$ is at most the sum of the entropies of H and the X_i's). The standard analysis mentioned above requires an additional $2\log(T/\epsilon)$ bits of entropy per block. In the randomness extraction literature, the additional entropy required is typically not significant because $\log(T/\epsilon)$ is much smaller than m. However, it can be significant in our applications. For example, a typical setting is hashing $T = \Theta(M)$ items into $2^m = M$ bins. Here $m + 2\log(T/\epsilon) \geq 3m - O(1)$ and thus the standard analysis requires 3 times more entropy than the lower bound of $(1 - o(1))m$. (The bounds obtained for the specific applications mentioned below are

[1] See Section 1.2.4 for a brief introduction to and formal definition of randomness extractors.

even larger, sometimes due to the need for a subconstant $\epsilon = o(1)$ and sometimes due to the fact that several independent hash values are needed for each item.)

By a finer analysis, the required entropy per block for $(H(X_1), \ldots, H(X_T))$ to be ϵ-close to uniform in statistical distance can be reduced from $m + 2\log(T/\epsilon)$ to $m + \log T + 2\log(1/\epsilon)$, which is known to be tight. The entropy required can be reduced even further for some applications by measuring the quality of the output differently (not using statistical distance) or by using 4-wise independent hash functions (which also have very fast implementations).

1.1.3 Applications

Consider the standard method of *chained hashing*, when T items are hashed into T buckets by a single random hash function. When the hash function is an idealized truly random function, the maximum load of any bucket is known to be $(1 + o(1)) \cdot (\log T / \log \log T)$ with high probability. In contrast, for a natural family of 2-universal hash functions, it is possible for an adversary to choose a set of T items so that the maximum load is always $\Omega(T^{1/2})$. The results of this chapter in turn show that 2-universal hashing achieves the same performance as ideal hashing asymptotically, provided that the data comes from a block source with roughly $2\log T$ bits of (Rényi) entropy per item. Similar results for other applications of hashing, such as "linear probing," "balanced allocations," and "Bloom filters," are described in Sections 1.6 and 1.7.

1.1.4 Perspective

The block source model we consider in this chapter is very much in the same spirit as the other semi-random models covered in this book, in that an adversary can pick a worst-case input distribution from a constrained family of distributions (namely, ones with enough conditional entropy per data item). In fact, the semi-random graph models discussed in Chapter **??** were also inspired by a model of "semi-random sources" in the randomness extractor literature, which amount to block sources where each block consists of one bit and inspired the later, more general notion of block sources we study. Moreover, the block source model itself (with respect to max probability) is identical to the class of distributions considered for "diffuse adversaries" in the online paging problem in Chapter **??**.

In terms of the goals for the analysis of algorithms laid out in Chapter **??**, the motivation for the model in the current chapter falls squarely under "performance prediction." Indeed, the goal is to understand when an instantiation of a hashing algorithm with an explicit and efficient family of hash functions will perform similarly to its idealized analysis with truly random functions. The design of an algorithm that optimizes the idealized performance is taken as a given, and the only freedom is in the choice of hash family to instantiate the algorithm.

The block source model seems relatively uncontroversial from the "natural scientist" perspective, where the goal is to explain why previous experiments have not witnessed the gap between universal hashing and idealized analysis predicted by the worst-case theory. Indeed, those experiments may have used input distributions that are block sources or interact with the hash functions in a similar way.

From the "engineering" perspective, where the goal is to select a hash family to achieve good performance in a new application, more care is warranted in judging whether the data distribution is likely to fit the block source model. Even if each data item has sufficient entropy on its own, correlations between data items can make the conditional entropies very small or even zero. An extreme example is when the items are consecutive elements of an interval, i.e. $x_{i+1} = x_i + 1$. If the initial data item x_1 has high entropy, so will all of the other items x_i, but the conditional entropy of x_i given $x_1, \ldots, x_{i-1}$ will be zero. Indeed, such data distributions do sometimes arise in practice, and practical 2-universal hashing families have been found to have poor performance (e.g. when used in linear probing) on such data distributions. Unfortunately, determining whether a distribution is close to a block source is difficult in general; it requires a number of samples that is exponential in the number of blocks (Exercise 1.5).

For this reason, an interesting direction for future work is to identify other families of data distributions (beyond just the block-source model) where simple families of hash functions can be shown to behave like idealized hash functions for a wide class of hashing applications. When that can't be done, there may be no alternative other than to turn to more complex hash families, such as $\omega(1)$-wise independence or cryptographic hash functions.

We remark that the cryptography literature also grapples with a similar issue around the implementation of cryptographic protocols that utilize hashing. It is often easier to provide the security of such protocols by adopting the "random oracle model," where the hash function is modelled as a truly random function that all parties (honest or adversarial) can query as an oracle. But security in the random oracle model does not in general imply security when the hash function is implemented with any explicit polynomial-time instantiation. Thus, there is a large body of work on trying to identify classes of cryptographic protocols and realistic, non-idealized properties of hash families such that the security in the random oracle model is preserved under instantiation (assuming that the hash family actually has the specified properties).

The cryptographic setting is usually much more challenging than the algorithmic one, because an adversary can typically influence the inputs fed to the hash function based on the description of the hash function itself, or at least based on prior observations of input-output behavior. In contrast, even in the worst-case analysis of hashing algorithms, we typically assume that the hash function is chosen randomly and independently of the data items. This assumption should be carefully scrutinized in applications of hashing algorithms (especially when used in

a security context), and if it does not hold, then cryptographic hash functions or "pseudorandom functions" may be a safer, albeit more expensive, choice.

1.2 Preliminaries

1.2.1 Notation

$[N]$ denotes the set $\{0, \ldots, N-1\}$. All logs are base 2. For a random variable X and an event E, $X|_E$ denotes X conditioned on E. The *support* of X is $\text{supp}(X) = \{x : \Pr[X = x] > 0\}$. For a finite set S, U_S denotes a random variable uniformly distributed on S.

1.2.2 Hashing

Let $\mathcal{H}$ be a family (multiset) of hash functions $h : [N] \to [M]$ and let H be uniformly distributed over $\mathcal{H}$. We use $h \leftarrow H$ to denote that h is sampled according to the distribution H. We say that $\mathcal{H}$ is a *truly random family* if $\mathcal{H}$ is the set all functions mapping $[N]$ to $[M]$, i.e. the N random variables $\{H(x)\}_{x\in[N]}$ are independent and uniformly distributed over $[M]$. For $s \in \mathbb{N}$, $\mathcal{H}$ is *s-wise independent* (a.k.a. *strongly s-universal*) if for every sequence of distinct elements $x_1, \ldots, x_s \in [N]$, the random variables $H(x_1), \ldots, H(x_s)$ are independent and uniformly distributed over $[M]$. $\mathcal{H}$ is *s-universal* if for every sequence of distinct elements $x_1, \ldots, x_s \in [N]$, we have $\Pr[H(x_1) = \cdots = H(x_s)] \leq 1/M^s$. The description size of $H \in \mathcal{H}$ is the number of bits to describe H, which is simply $\log |\mathcal{H}|$.

A standard example of a 2-universal family for a universe $[N]$ being hashed to the range $[M]$ can be obtained by choosing a prime $p \geq \max\{N, M\}$ and using the family

$$h_{a,b}(x) = ((ax + b) \bmod p) \bmod M.$$

Here a and b are integers chosen independently and uniformly at random from $\{1, \ldots, p-1\}$ and $\{0, 1, \ldots, p-1\}$, respectively. We leave it as Exercise 1.2 to show that for $x \neq y$, the probability that $h_{a,b}(x) = h_{a,b}(y)$ is at most $1/M$. A standard example of an s-wise independent family where the domain and range are the same finite field $\mathbb{F}$ is the family of hash functions

$$h_{a_0,a_1,\ldots,a_{s-1}}(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{s-1}. \tag{1.1}$$

Here choosing a hash function from the family corresponds to choosing the a_i's independently and uniformly at random from $\mathbb{F}$; that is, the hash function is a random polynomial of degree at most $s-1$. Again, we leave it as Exercise 1.3 to show that for a hash function chosen randomly from this family, the hashed values for any s elements of $\mathbb{F}$ are distributed uniformly and independently over $\mathbb{F}$. To

obtain a family where the range is smaller than the domain, we can use a field $\mathbb{F}$ of size p^k for a prime p and integer $k > 1$, pick any positive integer $\ell < k$, and compose the hash functions in Eq. (1.1) with a fixed mapping $g : \mathbb{F} \to [p^\ell]$, all of whose preimages are of size $\mathbb{F}/p^\ell$ (e.g. $g(y) = y \bmod p^\ell$, after interpreting y as an element of $[p^k]$).

1.2.3 Block Sources

We view the data items as being random variables distributed over a finite set of size N, which we identify with $[N]$. We use the following quantities to measure the amount of randomness in a data item. For a random variable X, the *max probability* of X is $\mathrm{mp}(X) = \max_x \Pr[X = x]$. The *collision probability* of X is $\mathrm{cp}(X) = \sum_x \Pr[X = x]^2$. Measuring these quantities is equivalent to measuring the *min-entropy*

$$\mathrm{H}_\infty(X) = \min_x \log(1/\Pr[X = x]) = \log(1/\mathrm{mp}(X))$$

and the *Rényi entropy*

$$\mathrm{H}_2(X) = \log(1/\Pr[X = X']) = \log(1/\mathrm{cp}(X)),$$

where X' is an i.i.d. copy of X. (These entropy measures are from the family of Rényi q-entropies, defined for positive and finite $q \neq 1$ as

$$\mathrm{H}_q(X) = \frac{1}{1-q} \log \left(\sum_x (\Pr[X = x])^q \right),$$

with $\mathrm{H}_\infty(X)$ obtained by taking the limit as $q \to \infty$, and Shannon entropy by taking the limit as $q \to 1$.) If X is supported on a set of size K, then $\mathrm{mp}(X) \geq \mathrm{cp}(X) \geq 1/K$ (i.e. $\mathrm{H}_\infty(X) \leq \mathrm{H}_2(X) \leq \log K$), with equality if and only if X is uniform on its support. Hence, assuming X has at least k bits of Rényi entropy is strictly weaker than assuming X has at least k bits of min-entropy, and therefore using Rényi entropy makes the positive results stronger. On the other hand, it also holds that $\mathrm{mp}(X) \leq \mathrm{cp}(X)^{1/2}$ (i.e. $\mathrm{H}_\infty(X) \geq \mathrm{H}_2(X)/2$; see Exercise 1.1), so min-entropy and Rényi entropy are always within a factor of 2 of each other.

We model a sequence of data items as a sequence $(X_1, \ldots, X_T)$ of correlated random variables where each item is guaranteed to have some entropy even conditioned on the previous items.

Definition 1.1 (Block Sources) A sequence of random variables $(X_1, \ldots, X_T)$ taking values in $[N]^T$ is a *block source with collision probability p per block* (respectively, *max probability p per block*) if for every $i \in [T]$ and every $(x_1, \ldots, x_{i-1}) \in \mathrm{supp}(X_1, ..., X_{i-1})$, we have $\mathrm{cp}(X_i|_{X_1=x_1,\ldots,X_{i-1}=x_{i-1}}) \leq p$ (respectively, $\mathrm{mp}(X_i|_{X_1=x_1,\ldots,X_{i-1}=x_{i-1}}) \leq p$).

When *max probability* is used as the measure of entropy, then this is a standard model of sources used in the randomness extractor literature. We will mainly use the *collision probability* formulation as the entropy measure, since it makes the statements more general.

Definition 1.2 $(X_1, \ldots, X_T)$ is a *block K-source* if it is a block source with collision probability at most $p = 1/K$ per block.

1.2.4 Randomness Extractors

Before obtaining our results on block sources, we need results showing that hashing a single item leads to it being nearly uniformly distributed, which we will then generalize. A *randomness extractor* can be viewed as a family of hash functions with the property that for any random variable X with enough entropy, if we pick a random hash function h from the family, then $h(X)$ is "close" to being uniformly distributed on the range of the hash function. Randomness extractors are a central object in the theory of pseudorandomness and have many applications in theoretical computer science. Thus there is a large body of work on the construction of randomness extractors. A major emphasis in this line of work is constructing extractors where it takes extremely few (e.g. a logarithmic number of) random bits to choose a hash function from the family. This parameter is less crucial for us, so instead our emphasis is on using simple and very efficient hash functions (e.g. universal hash functions) and minimizing the amount of entropy needed from the source X. To do this, we will measure the quality of a hash family in ways that are tailored to our application, and thus we do not necessarily work with the standard definitions of extractors.

In requiring that the hashed value $h(X)$ be 'close' to uniform, the standard definition of an extractor uses the most natural measure of 'closeness'. Specifically, for random variables X and Y, taking values in $[N]$, their *statistical distance* is defined as

$$\Delta(X, Y) = \max_{S \subseteq [N]} |\Pr[X \in S] - \Pr[Y \in S]|.$$

X and Y are called *ϵ-close* (resp., ϵ-far) if $\Delta(X, Y) \leq \epsilon$ (resp., $\Delta(X, Y) \geq \epsilon$).

The classic Leftover Hash Lemma shows that universal hash functions are randomness extractors with respect to statistical distance.

Lemma 1.3 (The Leftover Hash Lemma) *Let $H : [N] \to [M]$ be a random hash function from a 2-universal family $\mathcal{H}$. For every random variable X taking values in $[N]$ with $\mathrm{cp}(X) \leq 1/K$, we have*

$$\mathrm{cp}((H, H(X))) \leq \frac{1}{|\mathcal{H}|} \cdot \left(\frac{1}{M} + \frac{1}{K}\right),$$

and thus $(H, H(X))$ *is* $(1/2) \cdot \sqrt{M/K}$*-close to* $(H, U_{[M]})$.

Notice that the above lemma says that the *joint* distribution of $(H, H(X))$ is ϵ-close to uniform (for $\epsilon = (1/2) \cdot \sqrt{M/K}$); a family of hash functions achieving this property is referred to as a "strong" randomness extractor. Up to some loss in the parameter ϵ (which we will later want to save), this strong extraction property is equivalent to saying that with high probability over $h \leftarrow H$, the random variable $h(X)$ is close to uniform.

Proof Let (H', X') be an i.i.d. copy of (H, X). The bound on the collision probability follows by the following calculation.

$$\begin{aligned}
\mathrm{cp}(H, H(X)) &= \Pr[H = H' \wedge H(X) = H'(X')] \\
&= \Pr[H = H'] \cdot \Pr[H(X) = H(X')] \\
&\leq (1/|\mathcal{H}|) \cdot (\Pr[X = X'] + \Pr[H(X) = H(X') | X \neq X']) \\
&\leq \frac{1}{|\mathcal{H}|} \cdot \left(\frac{1}{M} + \frac{1}{K} \right).
\end{aligned}$$

The bound on the statistical distance follows directly by the second statement of the following lemma (we state the more general first statement since it will be useful later). □

Lemma 1.4 *If* X *takes values in* $[M]$ *and* $\mathrm{cp}(X) \leq 1/M + 1/K$*, then:*

1 For every function $f : [M] \to \mathbb{R}$*,*

$$|\mathrm{E}[f(X)] - \mu| \leq \sigma \cdot \sqrt{M/K},$$

where μ *is the expectation of* $f(U_{[M]})$ *and* σ *is its standard deviation. In particular, if* f *takes values in the interval* $[a, b]$*, then*

$$|\mathrm{E}[f(X)] - \mu| \leq \sqrt{(\mu - a) \cdot (b - \mu)} \cdot \sqrt{M/K}.$$

2 X *is* $(1/2) \cdot \sqrt{M/K}$*-close to* $U_{[M]}$.

Proof By the premise of the lemma,

$$\begin{aligned}
|\mathrm{E}[f(X)] - \mu| &= \left| \sum_{x \in [M]} (f(x) - \mu) \cdot (\Pr[X = x] - 1/M) \right| \\
&\leq \sqrt{\sum_{x \in [M]} (f(x) - \mu)^2} \cdot \sqrt{\sum_{x \in [M]} (\Pr[X = x] - 1/M)^2} \qquad \text{(Cauchy-Schwarz)} \\
&= \sqrt{M \cdot \mathrm{Var}[f(U_{[M]})]} \cdot \sqrt{\sum_{x \in [M]} (\Pr[X = x]^2 - 2\Pr[X = x]/M + 1/M^2)} \\
&= \sqrt{M} \cdot \sigma \cdot \sqrt{\mathrm{cp}(X) - 2/M + 1/M} \\
&\leq \sigma \cdot \sqrt{M/K}.
\end{aligned}$$

The "in particular" follows from the fact that $\sigma[Y] \leq \sqrt{(\mu - a) \cdot (b - \mu)}$ for every random variable Y taking values in $[a, b]$ and having expectation μ. (Intuitively, the variance is maximized if Y takes on only the extreme values a and b with appropriate probability masses to make the expectation μ. For a proof: $\sigma[Y]^2 = \mathrm{E}[(Y - a)^2] - (\mu - a)^2 \leq (b - a) \cdot (\mu - a) - (\mu - a)^2 = (\mu - a) \cdot (b - \mu)$.)

Item (2) follows by noting that the statistical distance between X and $U_{[M]}$ is the maximum of $|\mathrm{E}[f(X)] - \mathrm{E}[f(U_{[M]})]|$ over Boolean functions f, since a Boolean function can be viewed as an indicator/characteristic function for a subset S in the definition of statistical distance. Hence, by Item (1), the statistical distance is at most $\sqrt{\mu(f) \cdot (1 - \mu(f))} \cdot \sqrt{M/K} \leq (1/2) \cdot \sqrt{M/K}$. □

1.3 Hashing Block Sources

In this section, we show that when the data is modeled as a block source with sufficient entropy per block, the performance of using 2-universal hash functions is close to that of ideal hashing. More precisely, note that in ideal hashing, the distribution of the hashed values $(H(x_1), \ldots, H(x_T))$ is simply a uniform distribution over $[M]^T$. The following theorem states that when the data is a block source $(X_1, \ldots, X_T)$ with sufficient entropy per block and H is 2-universal, the distribution of the hashed values $(H(X_1), \ldots, H(X_T))$ is close to uniform in statistical distance.

Theorem 1.5 *Let $H : [N] \to [M]$ be a random hash function from a 2-universal family $\mathcal{H}$. For every block K-source $(X_1, \ldots, X_T)$, the random variable $(H, H(X_1), \ldots, H(X_T))$ is $(T/2) \cdot \sqrt{M/K}$-close to $(H, U_{[M]^T})$.*

Proof The theorem follows by applying the Leftover Hash Lemma (Lemma 1.3) to each block of the source and summing the statistical distance over the T blocks.

Specifically, since $(X_1, \ldots, X_T)$ is a block K-source, the Leftover Hash Lemma implies that for every $x_{<i} = (x_1, \ldots, x_{i-1}) \in [M]^{i-1}$, the distribution $(H, H(X_i)|_{X_{<i}=x_{<i}})$ is $(1/2) \cdot \sqrt{M/K}$-close to $(H, U_{[M]})$.

Define hybrid distributions $D_1, \ldots, D_{T+1}$ by

$$D_i = \left(H, H(X_1), \ldots, H(X_{i-1}), U_{[M]}^{(i)}, \ldots, U_{[M]}^{(T)}\right) \text{ for } i \in [T+1].$$

Note that the above statement implies that $\Delta(D_i, D_{i+1}) \leq (1/2) \cdot \sqrt{M/K}$ for every $i \in [T]$ (since the statement holds for every $x_{<i} \in [M]^{i-1}$). Also note that $D_1 = (H, U_{[M]^T})$ and $D_{T+1} = (H, H(X_1), \ldots, H(X_T))$. Since statistical distance satisfies the triangle inequality,

$$\Delta(D_1, D_{T+1}) \leq \sum_{i=1}^{T} \Delta(D_i, D_{i+1}) \leq (T/2) \cdot \sqrt{M/K}.$$

□

Assuming $K \geq MT^2/(4\epsilon^2)$, Theorem 1.5 implies that the distribution of the hashed values $(H(X_1), \ldots, H(X_T))$ is ϵ-close to uniform. Thus, any event that would occur with some probability p under ideal hashing now occurs with probability $p \pm \epsilon$. This allows us to readily translate existing results for ideal hashing into results for universal hashing in the block data source model.

A tight version of Theorem 1.5 is known, which improves the linear dependency in T to $\sqrt{T}$:

Theorem 1.6 *Let $H : [N] \to [M]$ be a random hash function from a 2-universal family $\mathcal{H}$. For every block K-source $(X_1, \ldots, X_T)$, the random variable $(H, H(X_1), \ldots, H(X_T))$ is $(\sqrt{MT/K})$-close to $(H, U_{[M]^T})$.*

By Theorem 1.6, it suffices to assume $K \geq MT/\epsilon^2$ to conclude that the hashed values are ϵ-close to uniform. Below we discuss the implication of Theorem 1.6 to chained hashing as an example.

The proof of Theorem 1.6 is quite involved, switching carefully between different distance notions to measure the growth of the distance to the uniform distribution over the T blocks. Thus we omit its proof here.

1.4 Application: Chained Hashing

We first briefly recall the chained hashing algorithm and its results under ideal hashing. A hash table using *chained hashing* stores a set $\overline{x} = \{x_1, \ldots, x_T\} \in [N]^T$ in an array of M buckets. Let h be a hash function mapping $[N]$ to $[M]$. We place each item x_i in the bucket $h(x_i)$. The *load* of a bucket when the process terminates is the number of items in it.

Definition 1.7 Given $h : [N] \to [M]$ and a sequence $\overline{x} = \{x_1, \ldots, x_T\}$ of data items from $[N]$ stored via chained hashing using h, we define the *maximum load* $\mathrm{MaxLoad}_{\mathrm{CH}}(\overline{x}, h)$ to be the maximum load among the buckets after all data items have been placed.

It is known that under ideal hashing, when $M = T$, the expected maximum load is $\log T / \log \log T$ asymptotically. This bound also holds with high probability. More precisely, we have:

Theorem 1.8 *Let H be a truly random hash function mapping $[N]$ to $[T]$. For every sequence $\overline{x} \in [N]^T$ of distinct data items, we have*

$$\Pr\left[\mathrm{MaxLoad}_{\mathrm{CH}}(\overline{x}, H) \leq (1 + o(1)) \cdot \frac{\log T}{\log \log T}\right] = 1 - o(1),$$

where the $o(1)$ terms tend to zero as $T \to \infty$.

The calculation underlying this theorem requires that the hash function be $\Omega(\log T / \log \log T)$-wise independent. Indeed, Exercise 1.4 shows that there are 2-universal families and input sets $\overline{x}$ where the maximum load is always at least $\sqrt{T}$. Nevertheless, suppose we model the data as a block source and use 2-universal hashing, Theorem 1.6 allows us to derive the conclusion of Theorem 1.8 assuming the data has sufficient entropy per block.

Theorem 1.9 *Let H be chosen at random from a 2-universal hash family $\mathcal{H}$ mapping $[N]$ to $[T]$. For every block K-source $\overline{X}$ taking values in $[N]^T$ with $K = \omega(T^2)$, we have*

$$\Pr\left[\mathrm{MaxLoad}_{\mathrm{CH}}(\overline{X}, H) \leq (1 + o(1)) \cdot \frac{\log T}{\log \log T}\right] = 1 - o(1),$$

where the $o(1)$ terms tend to zero as $T \to \infty$.

Proof Set $M = T$. Note that the value of $\mathrm{MaxLoad}_{\mathrm{CH}}(\overline{x}, h)$ can be determined from the hashed sequence $(h(x_1), \ldots, h(x_T)) \in [M]^T$ alone, and does not otherwise depend on the data sequence $\overline{x}$ or the hash function h. Thus for a function $\lambda : N \to N$, we can let $S \subseteq [M]^T$ be the set of all sequences of hashed values that produce an allocation with a max load greater than $\lambda(T)$. By Theorem 1.8, we can take $\lambda(T) = (1 + o(1)) \cdot (\log T)/(\log \log T)$ so that we have:

$$\Pr[U_{[M]^T} \in S] = \Pr\left[\mathrm{MaxLoad}_{\mathrm{CH}}(\overline{x}, I) > \lambda(T)\right] = o(1),$$

where I is a truly random hash function mapping $[N]$ to $[M] = [T]$ and $\overline{x}$ is an arbitrary sequence of distinct data items.

We are interested in the quantity:

$$\Pr[\mathrm{MaxLoad}_{\mathrm{CH}}(\overline{X}, H) > \lambda(T)] = \Pr[(H(X_1), \ldots, H(X_T)) \in S],$$

where H is a random hash function from a 2-universal family. Given $K = \omega(T^2)$,

we set $\epsilon = \sqrt{MT/K} = o(1)$. By Theorem 1.6, $(H(X_1), \ldots, H(X_T))$ is ϵ-close to the uniform distribution. Thus, we have

$$\begin{aligned}\Pr[(H(X_1), \ldots, H(X_T)) \in S] &\leq \Pr[U_{[M]^T} \in S] + \epsilon \\ &= o(1).\end{aligned}$$

□

1.5 Optimizing Block Source Extraction

We have seen that, for some applications, when the data comes from a block source with sufficient entropy per block, 2-universal hashing performs nearly as well as ideal hashing. We might naturally ask, how much entropy do we need? The answer can depend on the needs of the application scenario, as well as on the specific hashing algorithm we use and its analysis. With our analysis thus far, the required entropy can range from very reasonable to unrealistic.

In this section, we discuss some general approaches to reduce the required entropy. A main observation is that instead of working with the stringent notion of statistical distance, for several applications it suffices to ensure that the collection of hashed values $(H(X_1), \ldots, H(X_T))$ has (or is statistically close to having) sufficiently small collision probability, say within an $O(1)$ factor of that of the uniform distribution. Theorem 1.10 below provides a result of this form with smaller required entropy from the block source, which reduces required entropy for some applications (see Section 1.6).

Theorem 1.10 *Let $H : [N] \to [M]$ be a random hash function from a 2-universal family $\mathcal{H}$. For every block K-source $(X_1, \ldots, X_T)$ and every $\epsilon > 0$, the random variable $(H, \bar{Y}) = (H, H(X_1), \ldots, H(X_T))$ is ϵ-close to a distribution $(H, \bar{Z})$ with collision probability*

$$\mathrm{cp}(H, \bar{Z}) \leq \frac{1}{|\mathcal{H}| \cdot M^T} \cdot \left(1 + \frac{M}{\epsilon K}\right)^T.$$

In particular, if $K \geq MT/\epsilon$, then $(H, \bar{Z})$ has collision probability at most

$$\frac{1}{|\mathcal{H}| \cdot M^T} \cdot \left(1 + \frac{2MT}{\epsilon K}\right).$$

Note that the factor $1/(|\mathcal{H}| \cdot M^T)$ is the collision probability of the uniform distribution $(H, U_{[M]^T})$ and the factors $(1 + (M/\epsilon K))^T$ and $(1 + (2MT/\epsilon K))$ quantify the blow-up in collision probability as compared to ideal hashing. Also note that comparing to Theorem 1.6, the key savings is in the dependency on ϵ. Specifically, to achieve statistical distance ϵ, K only needs to be $\Omega(MT/\epsilon)$ as opposed

to $\Omega(MT/\epsilon^2)$. Thus, this is particularly useful in applications where $\epsilon = o(1)$ is required in the analysis.

We omit the proof of Theorem 1.10 here. Roughly, to prove Theorem 1.10, one first shows that a certain average form of the (conditional) collision probability for hashed blocks is small with high probability, using the Leftover Hash Lemma (Lemma 1.3) together with a Markov argument. Then, the theorem follows by showing that the distribution $(H, \bar{Z})$ can be obtained by carefully modifying the distribution of $(H, \bar{Y})$ based on the property established in the first step.

We can further improve the bound by using 4-wise independent hash functions, stated in the following theorem, where the dependency on ϵ is further improved. At a high level, the improvement comes from the fact that 4-wise independent hashing allows us to replace the Markov argument by Chebyshev's inequality in the proof.

Theorem 1.11 *Let $H : [N] \to [M]$ be a random hash function from a 4-wise independent family $\mathcal{H}$. For every block K-source $(X_1, \ldots, X_T)$ and every $\epsilon > 0$, the random variable $(H, \bar{Y}) = (H, H(X_1), \ldots, H(X_T))$ is ϵ-close to a distribution $(H, \bar{Z})$ with collision probability*

$$\mathrm{cp}(H, \bar{Z}) \leq \frac{1}{|\mathcal{H}| \cdot M^T} \left(1 + \frac{M}{K} + \sqrt{\frac{2M}{\epsilon K^2}}\right)^T .$$

In particular, if $K \geq MT + \sqrt{2MT^2/\epsilon}$, then $(H, \bar{Z})$ has collision probability at most $(1+\gamma)/(|\mathcal{H}| \cdot M^T)$, for $\gamma = 2 \cdot (MT + \sqrt{2MT^2/\epsilon})/K$.

1.6 Application: Linear Probing

In this section, we illustrate that Theorems 1.10 and 1.11 can be used to give a better analysis for some applications than using Theorem 1.6. We first mention that when applied to the "high-probability" statement in chained hashing, Theorem 1.10 and 1.11 do not yield significant improvement over Theorem 1.6. On the other hand, for applications where we only have bounds on expectations, applying Theorem 1.10 and 1.11 can reduce the required entropy. As we will see in the *linear probing* example below, this is because small $\epsilon = o(1)$ is required in the analysis.

We start with a brief review of linear probing (also covered in Chapter **??**). A hash table using linear probing stores a sequence $\overline{x} = (x_1, \ldots, x_T)$ of data items from $[N]$ using M memory locations. Given a hash function $h : [N] \to [M]$, we place the data items $x_1, \ldots, x_T$ sequentially as follows. The data item x_i first attempts placement at $h(x_i)$, and if this location is already filled, locations $(h(x_i)+1) \bmod M$, $(h(x_i)+2) \bmod M$, and so on are tried until an empty location is found. The ratio $\gamma = T/M$ is referred to as the *load* of the table. The efficiency of linear probing is measured by the insertion time for a new data item as a function of the load. (Other measures, such as the average time to search for items already in the table,

are also often studied, and the results here can be generalized to these measures as well.)

Definition 1.12 Given $h : [N] \to [M]$, a set $\overline{x} = \{x_1, \ldots, x_{T-1}\}$ of data items from $[N]$ stored via linear probing using h, and an extra data item $y \notin \bar{x}$, we define the *insertion time* $\mathrm{Time}_{\mathrm{LP}}(h, \overline{x}, y)$ to be the value of j such that y is placed at location $h(y) + (j-1) \bmod M$.

It is well-known that with ideal hashing (i.e., hashing using truly random hash functions), the expected insertion time can be bounded quite tightly as a function of the load.

Theorem 1.13 *Let H be a truly random hash function mapping $[N]$ to $[M]$. For every sequence $\overline{x} \in [N]^{T-1}$ and $y \notin \overline{x}$, we have $\mathrm{E}[\mathrm{Time}_{\mathrm{LP}}(H, \overline{x}, y)] \leq 1/(1-\gamma)^2$, where $\gamma = T/M$ is the load.*

It is known that the expected lookup time can be bounded in terms of γ (independent of T) using only $O(1)$-wise independence. Specifically, with 5-wise independence, the expected time for an insertion is $O(1/(1-\gamma)^{2.5})$ for any sequence $\overline{x}$. On the other hand, it is also known that there are examples of sequences $\overline{x}$ and pairwise independent hash families such that the expected time for a lookup is logarithmic in T (even though the load γ is independent of T).

Now, suppose we consider data items coming from a block K-source $(X_1, \ldots, X_{T-1}, X_T)$ where the item $Y = X_T$ to be inserted is the last block. An immediate application of Theorem 1.6, using just a 2-universal hash family, gives that if $K \geq MT/\epsilon^2$, the resulting distribution of the element hashes is ϵ-close to uniform. The effect of the ϵ statistical distance on the expected insertion time is at most ϵT, because the maximum insertion time is T. That is, if we let E_U be the expected time for an insertion when using a truly random hash function, and E_P be the expected time for an insertion using pairwise independent hash functions, we have

$$E_P \leq E_U + \epsilon T.$$

A natural choice is $\epsilon = o(1/T)$, so that the ϵT term is $o(1)$, giving that K needs to be $\omega(MT^3) = \omega(M^4)$ in the standard case where $T = \gamma M$ for a constant $\gamma \in (0,1)$ (which we assume henceforth). An alternative interpretation is that with probability $1-\epsilon$, our hash table behaves exactly as though a truly random hash function was used. In some applications, constant ϵ may be sufficient, in which case $K = \omega(M^2)$ suffices.

Better results can be obtained by applying Lemma 1.4, in conjunction with Theorem 1.10. In particular, for linear probing, the standard deviation σ of the insertion time is known to be $O(1/(1-\gamma)^2)$. With a 2-universal family, as long as $K \geq MT/\epsilon$, from Theorem 1.10 the resulting hash values are ϵ-close to a block source with collision probability at most $(1 + 2MT/(\epsilon K))/M^T$. We can now apply Lemma 1.4 with

this bound on the collision probability to bound the expected insertion time as

$$E_P \leq E_U + \epsilon T + \sigma\sqrt{\frac{2MT}{\epsilon K}}.$$

Choosing $\epsilon = o(1/T)$ gives that E_P and E_U are the same up to lower order terms when K is $\omega(M^3)$. Theorem 1.11 gives a further improvement; for $K \geq MT + \sqrt{\frac{2MT^2}{\epsilon}}$, we have

$$E_P \leq E_U + \epsilon T + \sigma\sqrt{\frac{2MT + \sqrt{\frac{2MT^2}{\epsilon}}}{K}}.$$

Choosing $\epsilon = o(1/T)$ now allows for K to be only $\omega(M^2)$.

In other words, the Rényi entropy needs only to be $2\log(M) + \omega(1)$ bits when using 4-wise independent hash functions, and $3\log(M) + \omega(1)$ bits for 2-universal hash functions. We formalize the result for the case of 2-universal hash functions as follows:

Theorem 1.14 *Let H be chosen at random from a 2-universal hash family $\mathcal{H}$ mapping $[N]$ to $[M]$. For every block K-source $(\overline{X}, Y)$ taking values in $[N]^T$ with $K \geq MT/\epsilon$, we have $\mathrm{E}[\mathrm{Time}_{\mathrm{LP}}(H, \overline{X}, Y)] \leq 1/(1-\gamma)^2 + \epsilon T + \sigma\sqrt{\frac{2MT}{\epsilon K}}$. Here $\gamma = T/M$ is the load and $\sigma = O(1/(1-\gamma)^2)$ is the standard deviation in the insertion time in the case of truly random hash functions.*

1.7 Other Applications

In this section, we survey the application of the methods in this chapter to a couple of other hashing-based algorithms.

With the *balanced allocation* paradigm, it is known that when T items are hashed to T buckets, with each item being sequentially placed in the least loaded of d choices (for $d \geq 2$), the maximum load is $\log\log T/\log d + O(1)$ with high probability. Note that going from $d = 1$ to $d = 2$ yields an exponential saving in the maximum load from $O(\log T/\log\log T)$ to $O(\log\log T)$. Using the methods from this chapter, it can be shown that the same result holds when the hash function is chosen from a 2-universal hash family, provided the data items come from a block source with roughly $(d+1)\log T$ bits of entropy per data item.

Bloom filters are data structures for approximately storing sets in which membership tests can result in false positives with some bounded probability. It can be shown that there is a constant gap in the false positive probability for worst-case data when $O(1)$-wise independent hash functions are used instead of truly random hash functions. On the other hand, if the data comes from a block source with roughly $3\log M$ bits of (Rényi) entropy per item, where M is the size of the Bloom

Type of Hash Family	Required Entropy
Linear Probing	
2-universal hashing	$3 \log T$
4-wise independence	$2 \log T$
Chained Hashing	
2-universal hashing	$2 \log T$
Balanced Allocations with d Choices	
2-universal hashing	$(d+1) \log T$
Bloom Filters	
2-universal hashing	$3 \log T$

Table 1.1 *Each entry denotes the (Rényi) entropy required per item to ensure that the performance of the given application is "close" to the performance when using truly random hash functions. In all cases, the bounds omit additive terms that depend on how close a performance is desired, and we restrict to the (standard) case that the size of the hash table is linear in the number of items being hashed. That is,* $m = \log T + O(1)$.

filter, then the false positive probability with 2-universal hashing asymptotically matches that of ideal hashing.

A summary of required (Rényi) entropy per item for the above applications can be found in Table 1.1.

1.8 Bibliographic Notes

The material in this chapter is drawn primarily from Chung et al. (2013).

Surveys of algorithmic applications of hashing are given by Knuth (1998), Broder and Mitzenmacher (2005), and Muthukrishnan (2005). Universal and s-wise independent hashing were introduced in Carter and Wegman (1979) and Wegman and Carter (1981), respectively. Analysis of $O(\log T)$-wise independent hashing of T items can be found in Schmidt and Siegel (1990); Pagh and Rodler (2004). Further coverage of universal and s-wise independent families can be found in many standard texts, e.g. Mitzenmacher and Upfal (2017) and Vadhan (2011). The analysis of maximum load under chained hashing with idealized hash functions (e.g., Theorem 1.8) was done by Gonnet (1981) and Raab and Steger (1998). Worst-case

analysis of chained hashing under 2-universal hash families (including Exercise 1.4) was done by Alon et al. (1999). The expected insertion time for linear probing under idealized hash functions (i.e. Theorem 1.13) was analyzed by Knuth (1998), and its variance can be found in Gonnet and Baeza-Yates (1991). Worst-case analysis of linear probing under $O(1)$-wise independence was done by Pagh et al. (2009) and Patrascu and Thorup (2010), with experiments (and fast implementations of 4-wise independent hashing) in Thorup and Zhang (2012). The balanced allocation paradigm is due to Azar et al. (2000) and Bloom filters are due to Bloom (1970). The fact that the performance of standard universal hashing often matches what is predicted for ideal hashing was experimentally observed in (Ramakrishna, 1989; Broder and Mitzenmacher, 2001; Dharmapurikar et al., 2004; Pagh and Rodler, 2004; Ramakrishna, 1988; Ramakrishna et al., 1997).

Surveys on randomness extractors are given by Nisan and Ta-Shma (1999), Shaltiel (2002), and Vadhan (2011). Block sources were introduced by Chor and Goldreich (1988) (under the name "probability-bounded sources"), generalizing the model of "semi-random sources" introduced by Santha and Vazirani (1986). The Leftover Hash Lemma (Lemma 1.3) is due to Bennett et al. (1988) Impagliazzo et al. (1989), with the proof we present being attributed to Rackoff (Impagliazzo and Zuckerman, 1989). The analysis of universal hashing and randomness extraction on block sources given by Theorem 1.5 follows Chor and Goldreich (1988) and Zuckerman (1996). Semi-random models for graphs were introduced by Blum and Spencer (1995) and diffuse adversaries for online paging by Koutsoupias and Papadimitriou (2000).

The Random Oracle Model in cryptography was introduced by Fiat and Shamir (1987) and Bellare and Rogaway (1993). The fact that security may not be preserved when instantiating the random oracle with explicit hash functions was demonstrated by Canetti et al. (2004). Efforts to find classes of protocols and properties of hash families that allow for secure instantiation can be found in Bellare et al. (2013) and the references therein.

References

Alon, Noga, Dietzfelbinger, Martin, Miltersen, Peter Bro, Petrank, Erez, and Tardos, Gábor. 1999. Linear hash functions. *Journal of the ACM*, **46**(5), 667–683.

Azar, Yossi, Broder, Andrei Z., Karlin, Anna R., and Upfal, Eli. 2000. Balanced Allocations. *SIAM Journal on Computing*, **29**(1), 180–200.

Bellare, Mihir, and Rogaway, Phillip. 1993. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. Pages 62–73 of: Denning, Dorothy E., Pyle, Raymond, Ganesan, Ravi, Sandhu, Ravi S., and Ashby, Victoria (eds), *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.* ACM.

Bellare, Mihir, Hoang, Viet Tung, and Keelveedhi, Sriram. 2013. Instantiating Random Oracles via UCEs. Pages 398–415 of: Canetti, Ran, and Garay, Juan A. (eds), *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Lecture Notes in Computer Science, vol. 8043. Springer.

Bennett, Charles H., Brassard, Gilles, and Robert, Jean-Marc. 1988. Privacy amplification by public discussion. *SIAM Journal on Computing*, **17**(2), 210–229. Special issue on cryptography.

Bloom, Burton H. 1970. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, **13**(7), 422–426.

Blum, Avrim, and Spencer, Joel. 1995. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, **19**(2), 204–234.

Broder, A., and Mitzenmacher, M. 2001. Using multiple hash functions to improve IP lookups. Pages 1454–1463 of: *INFOCOM 2001: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*.

Broder, A., and Mitzenmacher, M. 2005. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, **1**(4), 485–509.

Canetti, Ran, Goldreich, Oded, and Halevi, Shai. 2004. The random oracle methodology, revisited. *Journal of the ACM*, **51**(4), 557–594.

Carter, J. Lawrence, and Wegman, Mark N. 1979. Universal classes of hash functions. *Journal of Computer and System Sciences*, **18**(2), 143–154.

Chor, Benny, and Goldreich, Oded. 1988. Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity. *SIAM Journal on Computing*, **17**(2), 230–261.

Chung, Kai-Min, Mitzenmacher, Michael, and Vadhan, Salil P. 2013. Why Simple Hash Functions Work: Exploiting the Entropy in a Data Stream. *Theory of Computing*, **9**, 897–945.

Dharmapurikar, S., Krishnamurthy, P., Sproull, T.S., and Lockwood, J.W. 2004. Deep packet inspection using parallel bloom filters. *IEEE Micro*, **24**(1), 52–61.

Fiat, Amos, and Shamir, Adi. 1987. How to prove yourself: practical solutions to identification and signature problems. Pages 186–194 of: *Advances in cryptology—CRYPTO '86 (Santa Barbara, Calif., 1986)*. Lecture Notes in Comput. Sci., vol. 263. Springer, Berlin.

Gonnet, Gaston H. 1981. Expected Length of the Longest Probe Sequence in Hash Code Searching. *Journal of the ACM*, **28**(2), 289–304.

Gonnet, GH, and Baeza-Yates, R. 1991. *Handbook of algorithms and data structures: in Pascal and C*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Impagliazzo, Russell, and Zuckerman, David. 1989 (30 Oct.–1 Nov.). How to Recycle Random Bits. Pages 248–253 of: *30th Annual Symposium on Foundations of Computer Science*. IEEE, Research Triangle Park, North Carolina.

Impagliazzo, Russell, Levin, Leonid A., and Luby, Michael. 1989 (15–17 May). Pseudo-random Generation from one-way functions (Extended Abstracts). Pages 12–24 of: *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*.

Knuth, D.E. 1998. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA.

Koutsoupias, Elias, and Papadimitriou, Christos H. 2000. Beyond competitive analysis. *SIAM Journal on Computing*, **30**(1), 300–317.

Mitzenmacher, Michael, and Upfal, Eli. 2017. *Probability and computing*. Second edn. Cambridge University Press, Cambridge. Randomization and probabilistic techniques in algorithms and data analysis.

Muthukrishnan, S. 2005. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, **1**(2).

Nisan, Noam, and Ta-Shma, Amnon. 1999. Extracting Randomness: A Survey and New Constructions. *Journal of Computer and System Sciences*, **58**(1), 148–173.

Pagh, Anna, Pagh, Rasmus, and Ruzic, Milan. 2009. Linear Probing with Constant Independence. *SIAM J. Comput.*, **39**(3), 1107–1120.

Pagh, R., and Rodler, F.F. 2004. Cuckoo hashing. *Journal of Algorithms*, **51**(2), 122–144.

Patrascu, Mihai, and Thorup, Mikkel. 2010. On the k-Independence Required by Linear Probing and Minwise Independence. Pages 715–726 of: Abramsky, Samson, Gavoille, Cyril, Kirchner, Claude, Meyer auf der Heide, Friedhelm, and Spirakis, Paul G. (eds), *ICALP (1)*. Lecture Notes in Computer Science, vol. 6198. Springer.

Raab, Martin, and Steger, Angelika. 1998. "Balls into bins"—a simple and tight analysis. Pages 159–170 of: *Randomization and approximation techniques in computer science (Barcelona, 1998)*. Lecture Notes in Computer Science, vol. 1518. Berlin: Springer.

Ramakrishna, M. V. 1988. Hashing practice: analysis of hashing and universal hashing. Pages 191–199 of: *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM Press.

Ramakrishna, M. V. 1989. Practical performance of Bloom filters and parallel free-text searching. *Communications of the ACM*, **32**(10), 1237–1239.

Ramakrishna, M. V., Fu, E., and Bahcekapili, E. 1997. Efficient Hardware Hashing Functions for High Performance Computers. *IEEE Trans. Comput.*, **46**(12), 1378–1381.

Santha, Miklos, and Vazirani, Umesh V. 1986. Generating Quasi-random Sequences from Semi-random Sources. *Journal of Computer and System Sciences*, **33**(1), 75–87.

Schmidt, Jeanette P., and Siegel, Alan. 1990. The Analysis of Closed Hashing under Limited Randomness (Extended Abstract). Pages 224–234 of: *STOC*. ACM.

Shaltiel, Ronen. 2002. Recent Developments in Explicit Constructions of Extractors. *Bulletin of the European Association for Theoretical Computer Science*, **77**(June), 67–95.

Thorup, Mikkel, and Zhang, Yin. 2012. Tabulation-Based 5-Independent Hashing with Applications to Linear Probing and Second Moment Estimation. *SIAM J. Comput.*, **41**(2), 293–331.

Vadhan, Salil P. 2011. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, **7**(1-3), front matter, 1–336.

Wegman, Mark N., and Carter, J. Lawrence. 1981. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, **22**(3), 265–279.

Zuckerman, David. 1996. Simulating BPP Using a General Weak Random Source. *Algorithmica*, **16**(4/5), 367–391.

Exercises

1.1 Prove $\mathrm{H}_\infty(X) \geq \mathrm{H}_2(X)/2$ and find a distribution X such that the inequality is (almost) tight. (Hint: consider X that takes a fix value with a constant probability, say $1/2$, and is uniform for the rest of the probability mass.)

1.2 Prove that the family of hash functions of the form $h_{a,b}(x) = ((ax + b) \bmod p) \bmod M$ is a 2-universal family mapping $[N]$ to $[M]$ for $p \geq \max\{N, M\}$, $a \in \{1, \ldots, p-1\}$, and $b \in \{0, \ldots, p-1\}$.

1.3 Prove that for a finite field $\mathbb{F}$ and positive integer s, the family of functions of the form

$$h_{a_0,a_1,\ldots,a_{s-1}}(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{s-1}$$

is an s-wise independent family. (Hint: for every s distinct values $x_1, \ldots, x_s \in \mathbb{F}$ and every $y_1, \ldots, y_s \in \mathbb{F}$, there is a unique polynomial h of degree at most s such that $h(x_i) = y_i$ for all i.)

1.4 Let $\mathbb{F}$ be a finite field of size that is a perfect square, and let $\mathbb{F}_0 \subseteq \mathbb{F}$ be its subfield of size $\sqrt{|\mathbb{F}|}$. Consider the family $\mathcal{H}$ of hash functions mapping $\mathbb{F} \times \mathbb{F}$ to $\mathbb{F}$ given by $h_{a,b}(x, y) = ax + by$, where a, b vary over all of $\mathbb{F}$.

1. Show that $\mathcal{H}$ is a 2-universal family.
2. Let γ be a fixed element of $\mathbb{F} - \mathbb{F}_0$, and consider the set

$$S = \left\{ \left(\frac{1}{u+\gamma}, \frac{v}{u+\gamma} \right) : u, v \in \mathbb{F}_0 \right\}.$$

Observe that $|S| = |\mathbb{F}_0|^2 = |\mathbb{F}|$. Show that for every $a, b \in \mathbb{F}$, we have

$$\mathrm{MaxLoad}_{\mathrm{CH}}(S, h_{a,b}) \geq |\mathbb{F}_0| = \sqrt{|\mathbb{F}|}.$$

(Hint: when $b = 0$, consider the subset of S where $u = 0$. if $b \neq 0$, then argue that there is a $c \in \mathbb{F}$ such that c/b and $(\gamma c - a)/b$ are both in $\mathbb{F}_0$, and consider the subset of S where $v = (c/b)u + (\gamma c - a)/b$.)

1.5 Suppose that there is an algorithm A that takes as input s iid samples $X^{(1)}, \ldots, X^{(s)}$ from an unknown distribution X on $[N]^T$ and has the following properties:

- if X is the uniform distribution on $[N]^T$, then $A(X^{(1)}, \ldots, X^{(s)})$ accepts with probability at least $2/3$, and
- if X has statistical distance at least $1/2$ from every block source with collision probability at most $1/4$ per block, then $A(X^{(1)}, \ldots, X^{(s)})$ accepts with probability at most $1/3$.

Prove that the sample complexity s of A must be at least $\Omega(N^{(T-1)/2})$. (Hint: consider $X = (X_1, \ldots, X_T)$ where $X_1, \ldots, X_{T-1}$ are uniform and independent, and $X_T = f(X_1, \ldots, X_{T-1})$ for a randomly chosen function $f : [N]^{T-1} \to [N]$, and consider what must occur in the s samples for A to distinguish X from the uniform distribution on $[N]^T$.)

Acknowledgements

Kai-Min is supported in part by Academia Sinica Career Development Award Grant no. 23-17. Michael is supported by NSF grants CCF-1563710 and CCF-1535795. Salil is supported by NSF grant CCF-1763299 and a Simons Investigator Award. We thank Elias Koutsoupias and Tim Roughgarden for their feedback, which has improved this survey.