CS369N: Beyond Worst-Case Analysis
Lecture #6: Pseudorandom Data and Universal Hashing[*]

Tim Roughgarden[†]

November 25, 2009

1 Motivation: Linear Probing and Universal Hashing

This lecture discusses a very neat paper of Mitzenmacher and Vadhan [?], which proposes a robust measure of "sufficiently random data" and notes interesting consequences for hashing and some related applications.

We consider hash functions from $N = \{0,1\}^n$ to $M = \{0,1\}^m$, with $m < n$. We abuse notation and use N, M to denote both the sets and the cardinalities of the sets. Since a hash function $h : N \to M$ is effectively compressing a larger set into a smaller one, collisions (distinct elements $x, y \in N$ with $h(x) = h(y)$) are inevitable. A way of resolving collisions that is common in practice is *linear probing*, where given a data element x, one starts at the slot $h(x)$, and then proceeds to $h(x)+1$, $h(x)+2$, etc. until a suitable slot is found. (Either an empty slot if the goal is to insert x; or a slot that contains x if the goal is to search for x.) The linear search wraps around the table (from $M-1$ back to 0), if needed.

Recall that every fixed hash function performs badly on some data set, since by the Pigeonhole Principle there is a large data set of elements with equal hash values. Thus the analysis of hashing always involves some kind of randomization, either in the input or in the hash function. In some sense, the best-case scenario would be to have the hash function h that you use spread out the data set $S \subseteq N$ that you care about as evenly as possible, say as if each hash value $h(x)$ was an independent and uniform random sample from M. How could this "perfectly random" scenario arise?

1. A fixed hash function that maps N/M elements to each slot of M (e.g., using the low-order bits) and a uniformly random data set S. This is clearly an overly strong assumption on the data.

[*]©2009, Tim Roughgarden.

[†]Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: `tim@cs.stanford.edu`.

2. An arbitrary (worst-case) data set S and uniformly random hash function $h : N \to M$. The problem here is that a uniformly random function is too big to store and too slow to evaluate to be useful (in most applications), as you should think through.

The hope then, is that there are simpler and practical families of hash functions that guarantee the "spread" of worst-case data that is as good as uniformly random hashing. You should have learned the following definition in an undergraduate data structures and algorithms class.

Definition 1.1 (Universal Hash Function) A set $\mathcal{H}$ of hash functions from N to M is *(2-)universal* if for all distinct $x, y \in N$,

$$\mathbf{Pr}_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{M}. \tag{1}$$

Note the uniformly random hashing corresponding to taking $\mathcal{H}$ to be the set of all functions, in which case (1) holds with equality for every pair of distinct $x, y \in N$.

You should have learned as an undergraduate that a number of small and easy to evaluate families of universal hash functions exist (see e.g. [**?**]). Thus, it would be very cool if universal hash functions performed as well as perfectly random functions. You probably learned as an undergraduate that if collisions are resolved by chaining — where in each slot $m \in M$ one maintains a linked list of all elements with hash value m — that key performance metrics like expected search time are as good under universal hashing as under perfectly random hashing (see e.g. [**?**]).

The plot it thicker for linear probing, however.

1. Knuth [**?**] proved that with a perfectly random hash function, the expected insertion time of the Tth element under linear probing is $\approx \frac{1}{(1-\alpha)^2}$, where $\alpha = \frac{T}{m} < 1$ is the load of the hash table. Note that this guarantee depends only on the load α and is independent of the data set and hash table sizes.[1]

2. Pagh, Pagh, and Ruzia [**?**] showed that there exists a sequence of universal hash families and of data sets or arbitrarily large size T such that the expected insertion time under linear probing grows like $\Omega(\log T)$, even though the load $\alpha = \frac{T}{M}$ remains a fixed constant.

Note that the first result is far from obvious: even with perfectly random hashing, one might be worried about linear probing because of "clumps" — several consecutive full slots. Initially some clumps will arise just by random chance (analogous to the Birthday Paradox), and the concern is that linear probing exacerbates clump growth (when a new element hashes into a clump, it is guaranteed to increase the size of that clump). Knuth's analysis shows that this problem does occur to some extent (one can obtain better expected search time with more complicated re-probing strategies) but, with perfectly random hashing, the problem

[1]See Knuth [**?**] for his description of how this analysis of linear probing seduced him into a career in the design and analysis of algorithms!

remains controllable by controlling the load. The Pagh et al. [?] result demonstrates that, unlike for chaining, this type of guarantee does not hold for an arbitrary universal family of hash functions.

Experimentally, however, the performance of linear probing with a sensible choice of a universal hash function typically conforms to Knuth's idealized analysis. What really seems to be going on is that some degree of randomness in the data composes with a limited amount of randomness in the hash function to produce effectively perfectly random has values (Figure **??**). It is our duty as theorists to produce a satisfying and rigorous explanation.

Target Theorem: *"Sufficiently random data sets" and universal hashing perform "as well as" perfectly random hashing.*

2 Some Preliminaries

To make the target theorem a precise mathematical goal, we need to formalize the two phrases in quotes. We begin with the second phrase, where we use the standard and strong notion of proximity under the statistical distance measure.

Consider two distributions D_1, D_2 on a finite set X. The *statistical distance* is defined as

$$\max_{A \subseteq X} \left| \mathbf{Pr}_{D_1}[A] - \mathbf{Pr}_{D_2}[A] \right| .$$

Note that given D_1, D_2 it is easy to which which A maximizes this difference in probabilities: take A as the set of all $x \in X$ for which $\mathbf{Pr}_{D_1}[x] > \mathbf{Pr}_{D_2}[x]$. (Or, the complement of this set works just as well.) Note this also implies that the statistical distance between D_1, D_2 equals half of the ℓ_1 distance $\|D_1 - D_2\|_1$, when D_1, D_2 are viewed as vectors indexed by X. Finally, we say that D_1, D_2 are *ϵ-close* if the statistical distance between them is at most ϵ. Generally, if two distributions are close for a small value of ϵ, whatever analysis you do with respect to one of them (e.g., an idealized distribution) will carry over without much loss to the other (e.g., a realizable approximation of the idealized one). ϵ-closeness will be our precise notion of "performs as well as" in the Target Theorem.

Before considering a definition of "sufficiently random data set", we focus on a single element. Let X be a random variable with range N. We define the collision probability $\mathrm{cp}(X)$ of X as

$$\mathrm{cp}(X) = \sum_{i \in N} \mathbf{Pr}[X = i]^2 ,$$

which is the probability that the two independent copies of X take on the same value. Intuitively, a low collision probability would seem to force a random variable to be "pretty spread out" over its range. The requirement is weaker that restricting every probability $\mathbf{Pr}[X = i]$ to be small:

$$\mathrm{cp}(X) \le \max_{i \in N} \mathbf{Pr}[X = i] ,$$

since

$$\sum_{i \in N} \mathbf{Pr}[X = i]^2 \le \sum_{i \in N} \mathbf{Pr}[X = i] \cdot \max_{j \in N} \mathbf{Pr}[X = j] = \max_{j \in N} \mathbf{Pr}[X = j] .$$

As a canonical example, you should think about a random variable X that is uniform over some subset $T \subseteq N$ of the universe of size K, possibly much smaller than N. In this case, $\mathrm{cp}(X) = \max_{i \in N} \mathbf{Pr}[X = i] = 1/K$. For concreteness, you might want to think about the case where N is exponential in M, and K is polynomial (cubic, say) in M. Intuitively, if you could design a hash function h with advance knowledge of the set K, then obtaining perfect randomness over M would be easy (since the data is perfectly random over K). But what if you only know that such a K exists, and have to design a hash function that works well no matter which set it is? Note that with our example parameters, there are an exponential number of possible choices for the set K.

Our final definition of a "sufficiently random data set" corresponds to what's called a *block source* [?] in the derandomization literature. It is a sequence of random variables such that every random variable in the sequence always has low collision probability, no matter how the earlier random variables were instantiated.[2]

Definition 2.1 (Block Source with Entropy k [?]) A *block source with entropy k* is a sequence $X_1, \ldots, X_T$ of random variables on N such that, for every i and conditioned arbitrarily on $X_1, \ldots, X_{i-1}$,

$$\mathrm{cp}(X_i) \leq \frac{1}{2^k}.$$

3 Main Result and Discussion

The main result of this lecture is the following.

Theorem 3.1 ([?, ?, ?]) *Let $X_1, \ldots, X_T$ be a block source with range N and entropy at least k. Let $\mathcal{H}$ be a family of universal hash functions mapping N to M and h a random sample from $\mathcal{H}$. Then the joint distribution $(h, h(X_1), \ldots, h(X_T))$ is ϵ-close to the uniform distribution on $\mathcal{H} \times M^T$, where*

$$\epsilon = \frac{T}{2}\sqrt{\frac{M}{K}}$$

and $K = 2^k$.

The guarantee in Theorem 3.1 is an interesting one. To see this, consider fixing a target ϵ. To achieve this desired closeness, one needs

$$K \geq \frac{MT^2}{4\epsilon^2} \tag{2}$$

and hence the entropy k to satisfy

$$k \geq \log_2 M + 2\log_2 T + 2\log_2 \frac{1}{\epsilon} + O(1). \tag{3}$$

[2]Note this is essentially identical to the definition of a diffuse adversary in Lecture #2, except there the requirement of low collision probability was replaced by the more stringent condition that $\max_{i \in N} \mathbf{Pr}[X = i]$ is small.

Recall our canonical example of data elements drawn uniformly at random from a small "secret" set K, and with N exponential in M. Note that in the linear probing application, one would always have $T < M$, as otherwise the hash table overflows. Thus (2) states that as long as the secret set K has size at least a constant times a cubic function of M, a randomly chosen universal hash function will "iron out" the distribution so that it's essentially perfectly random over the slots (no matter which of the exponential candidates is the secret set).

The guarantee in Theorem 3.1 can be improved, but not by a lot. Passing from a single random variable to a sequence in a way more sophisticated than a naive induction (see Theorem 4.1 below) reduces the T^2 factor in (2) to a T [?], which in our canonical example means that universal hash functions iron out all quadratic-size secret sets. If the measure of "performs as well as" is relaxed from closeness to the uniform distribution to closeness to some distribution with small collision probability (which might be good enough in most hashing applications), then the factor of $\frac{1}{\epsilon^2}$ in (2) can be replaced by $\frac{1}{\epsilon}$ [?]. These dependencies cannot be reduced further [?], although in some cases assuming more than universality (i.e., t-wise independence for large enough t) enables improved upper bounds [?].

Finally, it might seem weird to phrase Theorem 3.1 in terms of the joint distribution $(h, h(X_1), \ldots, h(X_T))$, rather than the one $(h(X_1), \ldots, h(X_T))$ that we actually care about. There are a few reasons for this. First, the guarantee in Theorem 3.1 is stronger, which is always a plus. (The former implies that the $h(X_i)$'s are essentially uniform even after conditioning on the choice of h, while the latter does not.) Second, this stronger type of guarantee lends itself to induction arguments (see Theorem 4.1 below). Third, the tight lower bounds in [?] are for this stronger statement. The lower bounds on the data entropy required for the distribution $(h(X_1), \ldots, h(X_T))$ to be close to uniform are necessarily weaker — intuitively, the hash function can loan its randomness out to the data elements — although the gap is often not too large (see [?]).

4 The Leftover Hash Lemma

The proof of Theorem 3.1 boils down to what is known as the "Leftover Hash Lemma".

Theorem 4.1 (Leftover Hash Lemma [?]) *Let X be a random variable with range N and collision probability at most $1/K$. Let $\mathcal{H}$ be a family of universal hash functions mapping N to M and h a random sample from $\mathcal{H}$. Then the joint distribution $(h, h(X))$ is ϵ-close to the uniform distribution on $\mathcal{H} \times M$, where*

$$\epsilon = \frac{1}{2}\sqrt{\frac{M}{K}}.$$

Observe that this is the special case of Theorem 3.1 in which $T = 1$. It can be extended to the case of general T (with loss linear in T) by induction; we leave the (slightly tricky) details to Homework #3.

The proof of Theorem 4.1 involves some calculations but overall is quite slick. We first invoke the two hypothesis to upper bound the collision probability of the random variable $(H, H(X))$. From this we derive an upper bound on the ℓ_2 distance between its distribution and that of the uniform distribution, from which a bound on the statistical distance easily follows.

Recall that $\text{cp}(H, H(X))$ is the probability that two independent random samples (say $(h, h(x))$ and $(h', h'(x'))$ take on the same value. For this to happen, it must be that $h = h'$ and, furthermore, either $x = x'$ or x, x' collide under h. Thus,

$$\begin{aligned}\text{cp}(H, H(X)) &= \mathbf{Pr}_{x,x' \in N, h, h' \in \mathcal{H}}[(h, h(x)) = (h', h'(x'))] \\ &= \underbrace{\mathbf{Pr}[h = h']}_{1/|\mathcal{H}| \text{ since } h,h' \text{ u.a.r.}} \cdot \mathbf{Pr}[h(x) = h'(x') \ : \ h = h'] \, ;\end{aligned}$$

since

$$\mathbf{Pr}[h(x) = h'(x') \ : \ h = h'] = \underbrace{\mathbf{Pr}[x = x' \ : \ h = h']}_{= \text{cp}(X) \le 1/K} + \underbrace{\mathbf{Pr}[h(x) = h'(x') \ : \ h = h', x \neq x']}_{\le 1/M \text{ by universality}},$$

we conclude that

$$\text{cp}(H, H(X)) \le \frac{1}{|\mathcal{H}|}\left(\frac{1}{K} + \frac{1}{M}\right).$$

For comparison, the collision probability of the uniform distribution on $\mathcal{H} \times M$ would be $1/|\mathcal{H}|M$.

Next we need to translate this upper bound on collision probability into an upper bound on distance to the uniform distribution on $\mathcal{H} \times M$. We begin with a bound on ℓ_2 distance. Consider first two distributions p, q on an abstract finite set X. We note by p_x, q_x the corresponding probability masses on a point $x \in X$ Computing, we have

$$\|p - q\|_2^2 = \sum_{x \in X} (p_x - q_x)^2 = \sum_x p_x^2 + \sum_x q_x^2 - 2 \sum_x p_x q_x.$$

Now take q to be uniform on X. Then

$$\|p - q\|_2^2 = \sum_x p_x^2 + |\mathcal{H}|M \cdot \frac{1}{|H|^2 M^2} - \frac{2}{|\mathcal{H}|M} \underbrace{\sum_x p_x}_{=1} = \text{cp}(P) - \frac{1}{|\mathcal{H}|M},$$

where P denotes a random variable on X with distribution p. Plugging this bound into our context, we find that the squared ℓ_2 distance between the joint distribution on $(H, H(X))$ and of the uniform distribution on $\mathcal{H} \times M$ is at most

$$\frac{1}{|\mathcal{H}|}\left(\frac{1}{K} + \frac{1}{M}\right) - \frac{1}{|\mathcal{H}|M} = \frac{1}{|\mathcal{H}|K},$$

and the ℓ_2 distance is the square root of this. Finally, recall that for every vector $v \in \mathcal{R}^d$, we have $\|v\|_1 \le \sqrt{d} \cdot \|v\|_2$. (The all 1's vector is a tight example.) Since for us $d = |\mathcal{H}|M$ and since statistical distance is half of the ℓ_1 norm, we find that the statistical distance between the joint distribution on $(H, H(X))$ and the uniform distribution on $\mathcal{H} \times M$ is at most

$$\frac{1}{2} \cdot \sqrt{|\mathcal{H}|M} \cdot \sqrt{\frac{1}{|\mathcal{H}|K}} = \frac{1}{2}\sqrt{\frac{M}{K}},$$

as claimed.

5 Two More Applications

We conclude the lecture with two further uses of pseudorandom data and universal hashing. There will not be any serious content here, but these are topics that every theoretical computer scientist should have basic literacy in.

5.1 The Power of Two Choices

A simple but powerful approach to balancing load across resources is to use randomization. As an abstraction, think about throwing n balls independently and uniformly at random into n bins. A bread-and-butter application of the Chernoff bound shows that the expectation of the maximum number of balls in a single bin is $\Theta(\log n / \log\log n)$ (e.g. [?]). (This statistic is also clearly relevant for hashing, e.g. as the expected worst-case search time with chaining.) This upper bound is OK, but a little disappointing given that the expected load in any given bin is only 1.

An optimization is to throw the balls in sequentially and, for each ball, to randomly probe two different bins and throw the ball into the bin that is currently less populated (breaking ties randomly). Remarkably, this simple augmentation yields an exponential improvement, with the expected maximum population size now only $\Theta(\log\log n)$ [?, ?].[3]

The above analysis assumes that every ball is tossed uniformly and independently of all the others. What if the distribution on bins is instead generated by a random universal hash function? (Here, one assumes that balls have names and that there are two different hash functions that each maps names to bins.) This is a well-motivated question in some randomized load balancing applications, but it is unknown whether the $O(\log\log n)$ bound on the expected maximum population size continues to hold. Theorem 3.1 easily implies that as long as there is sufficient randomness in the ball names — that is, they come from a block source with entropy k satisfying (3), where M is the number of bins and T is the number of balls (canonically, $T = M$) — then the analyses for perfectly random bin choices continue to apply.

[3]One might hope that having *three* choices would reduce the expected maximum to $O(\log\log\log n)$; in fact, for $d \geq 2$, d choices only reduces the expected maximum population size to $\Theta(\log\log n / \log d)$ [?].

5.2 Bloom Filters

The basic bloom filter is a data structure that supports fast membership queries in applications that do not require deletions and that are tolerant of false positives. The basic scheme is to use ℓ different hash functions, with each hash function mapping the universe N to a set of M/ℓ slots (each of which contains only a single bit). These ℓ sets of slots are disjoint, so there are M slots overall. Initially, all bits are set to 0. When an item $x \in N$ is inserted, for each $i = 1, 2, \ldots, \ell$, the $h_i(x)$th slot of the ith group has its bit set to 1. (The bit may have already been set to 1 earlier, of course.) When an item $x \in N$ is searched for, the bloom filter reports a successful search if and only if, for each $i = 1, 2, \ldots, \ell$, the bit in the $h_i(x)$th slot of the ith group is set to 1.

Observe that there a bloom filter (with no deletions) never has a false negative: if x is inserted an later searched for, it will be found. There can, however, be false negatives: even if x was never inserted, other insertions can cause all of the corresponding ℓ bits to be set to 1, leading the bloom filter to believe that x *was* inserted at some point in the past. If we assume perfectly random hashing, then it's easy to estimate the probability that a given uninserted element y will suffer from a false positive after T insertions of other elements:

$$\left(1 - \left(1 - \frac{\ell}{M}\right)^T\right)^\ell \approx \left(1 - e^{\ell T/M}\right)^\ell.$$

For example, one can obtain a false positive probability of $2^{-\ell}$ when $\ell \approx \frac{M}{T} \ln 2$.

A true but non-obvious fact is that, assuming only universal hashing, strictly more space (i.e., a larger ℓ) is needed to obtain a given false positive probability [**?**]. Once again, Theorem 3.1 implies that, provided the data is sufficiently random along the lines of (3), the performance of universal hashing will be as good as that suggested by the idealized analysis above for perfectly random hashing.