CS 388R: Randomized Algorithms, Fall 2023 September 20th, 2023

Lecture 9: Limited Independence

Prof. Eric Price *Scribe: Alexia Atsidakou*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

In the previous lectures we discussed hash functions $h : U \to [m]$ that are uniformly random. This implies that it takes $U \log m$ space to remember the functions, which can be bigger than the hash table itself. This issue can be addressed using Limited Independence. However, the latter is not always a better choice. Below we mention some cases where each option is preferable.

Pros of Full Independence	Pros of Limited Independence
In some cases you don't need to remember the hash functions e.g. in load balancing (I don't care which machine runs each job, as long as it it scheduled).	For hash tables, you can actually store the hash function.
Hash functions may approximate fully random case: (i) if inputs have sufficiently entropy, (ii) if we have cryptographic hash functions. If so, full independence is a better model of the behavior.	If we have k-wise independence, i.e. sets of size k behave as if fully independent, then let $X_i =$ # elements in bin i. Then, suppose we store n elements. We have that $$\mathbb{E}\left(X_i^k\right) = \mathbb{E}\left(\left(\sum_{j=1}^{n} \mathbb{1}\{h(j) = i\}\right)^k\right)$$ only depends on k variables at a time. So, if we can bound the k-th moment we can get concentration guarantees (by Chebyshev).
By standard balls and bins analysis: $$\mathbb{E}\left(\max X_i\right) = \log n / \log \log n$$	Obtained guarantees are of order: $$\max X_i \leq n^{1/k}$$

2 Definitions

We begin with some definitions of limited independence.

Definition 1 (Universality)**.** *A family of hash functions $H = \{h : U \to [m]\}$ is called universal if $\forall x, y \in U$ with $x \neq y$,*

$$\mathbb{P}\left(h(x) = h(y)\right) \leq 1/m.$$

Family H is called ϵ-approximately universal if $\forall x, y \in U$ with $x \neq y$,

$$\mathbb{P}\left(h(x) = h(y)\right) \leq (1+\epsilon)/m.$$

Bound query time with universality. We show that universality suffices to bound the expected query time in hashtables. Suppose we hash a set S of n items to $[m]$, then

$$\begin{aligned}
\mathbb{E}\left(\text{time to query some key } x\right) &\leq \mathbb{E}\left(\#\text{ items } y \in S \text{ s.t. } h(y) = h(x)\ +1\right) \\
&\leq \sum_{y \in S} \mathbb{P}\left(h(y) = h(x)\right) + 1 \\
&\leq n\frac{1+\epsilon}{m} + 1,
\end{aligned}$$

which is $O(1)$ for $n = O(m)$.

Definition 2 (Pairwise Independence)**.** *A family of hash functions $H = \{h : U \to [m]\}$ is called pairwise independent if $\forall x, y \in U$ with $x \neq y$ and $s, t \in [m]$,*

$$\mathbb{P}\left(h(x) = s \cap h(y) = t\right) \leq 1/m^2.$$

We call H ϵ-approximately pairwise independent if

$$\mathbb{P}\left(h(x) = s \cap h(y) = t\right) \leq (1+\epsilon)/m^2.$$

Definition 3 (k-wise Independence)**.** *A family of hash functions $H = \{h : U \to [m]\}$ is k-wise independent if $\forall x_1, \ldots, x_k \in U$ with every $x_i \neq x_j$ and $s_1, \ldots, s_k \in [m]$,*

$$\mathbb{P}\left(h(x_1) = s_1 \cap \cdots \cap h(x_k) = s_k\right) \leq 1/m^k.$$

A family H is called ϵ-approximately k-wise independent if

$$\mathbb{P}\left(h(x_1) = s_1 \cap \cdots \cap h(x_k) = s_k\right) \leq (1+\epsilon)/m^k.$$

Example. For example, if $U = [3]$, the following family of functions is 2-wise independent but not 3-wise:

$$\begin{aligned}
&h(1) = a, \text{ where } a \text{ is randomly chosen from } [m] \\
&h(2) = b, \text{ where } b \text{ is randomly chosen from } [m] \\
&h(3) = a + b.
\end{aligned}$$

If we want to store n items into S bins, we have that $X_i = \sum_{j=1}^{n} \mathbb{1}\{h(j) = i\}$ and

$$\mathbb{E}\left(X_i^k\right) = \mathbb{E}\left(\left(\sum_{j=1}^{n} \mathbb{1}\{h(j) = i\}\right)^k\right).$$

Then, if we have k-wise independence and since the above variable only depends on k variables at each time, we can obtain concentration. Thus we can repeat the analysis as in the fully independent case.

Pairwise independence $\Rightarrow$ Universality. Below we show that Pairwise independence implies Universality. This holds because

$$\mathbb{P}\left(h(x)=h(y)\right)=\sum_{s\in[m]}\mathbb{P}\left(h(x)=s=h(y)\right)\leq\sum_{s\in[m]}\frac{1+\epsilon}{m^2}=\frac{1+\epsilon}{m}.$$

3 Examples

The goal is to construct Universal or Pairwise independent hash families H, where H can be written down and evaluated quickly. Some examples are presented below.

3.1 Example 1: Carter Wegman Hash Family.

Pick $P>U$ and select $a,b\in P$ uniformly. We have $h_{a,b}(x)=(ax+b) \mod P \mod m$. This family is $\frac{m}{P}$-approximately 2-wise independent. Generalizing this, the family $h_{a_1,\ldots,a_k}(x)=\left(\sum_{j=0}^{k}a_ix^i\right) \mod P \mod m$ is $\frac{m}{P}$-approximately k-wise independent.

Note. To store the above family we need to store only $a$and$b$, that is $2u=2\log U$ bits.

3.2 Example 2

Let the universe be $U=2^u$ and let $M=2^m$. We define

$$h_a(x)=(ax \mod U)>>(u-m),$$

where $>>$ denotes the right shift and a is a random odd number in $[U]$. This family is 2-approximately universal (proof left as exercise).

Note. To store the above family we need to store only a, that is $u=\log U$ bits.

3.3 Example 3

Let A be a $m\times u$ bit matrix, x is a bit vector of length u, and b is a bit vector of length m. We consider

$$h_{A,b}(x)=Ax+b.$$

Claim 4. *The above family is universal.*

Proof. For elements $x\neq y$ we have

$$\mathbb{P}\left(h_{A,b}(x)=h_{A,b}(y)\right)=\mathbb{P}\left(Ax+b=Ay+b\right)=\mathbb{P}\left(A(x-y)=0\right).$$

Now, we know that $x-y \neq 0$. Therefore, there must exist some coordinate j such that $(x-y)_j = 1$. Regardless of columns $1, \ldots, j-1, j+1, \ldots, u$ of A,

$$\mathbb{P}\left(A(x-y)=0\right) = \mathbb{P}\left(A_j = (A-A_j)(x-y)\right) = 1/2^m = 1/M$$

where $(A - A_j)$ denotes (abusing notation) matrix A with zeros in column j. The second to last inequality is because A_j is random and the right hand side, $(A-A_j)(x-y)$, is some fixed bit vector. □

Claim 5. *The above family is pairwise independent.*

Proof. We have that

$$\begin{aligned}\mathbb{P}\left(h_{A,b}(x)=\alpha, h_{A,b}(y)=\beta\right) &= \mathbb{P}\left(Ax+b=\alpha, Ay+b=\beta\right)\\ &= \mathbb{P}\left(Ax+b=\alpha, A(y-x)=(\beta-\alpha)\right)\\ &= \mathbb{P}\left(A(y-x)=(\beta-\alpha)\right)\cdot\mathbb{P}\left(Ax+b=\alpha | A(y-x)=(\beta-\alpha)\right).\end{aligned}$$

Now, we have $\mathbb{P}\left(A(y-x)=(\beta-\alpha)\right) \leq 1/2^m = 1/M$ as previously. For the second term, $A(y-x) = (\beta-\alpha)$ only depends on A. Therefore, regardless of the value of A, since b is also random, we have again $\mathbb{P}\left(Ax+b=\alpha | A(y-x)=(\beta-\alpha)\right) \leq 1/2^m = 1/M$. □

Note. In order to store the above functions we need to store A and b. Thus, we need $O(mu) = O(\log M \log U) \leq O(\log^2 U)$ bits. In fact, it suffices to use a matrix A that is Toeplitz (i.e. each row is a shift of the previous by 1 and the empty spots can be chosen randomly), which further reduces the number of bits required.

4 Perfect Hashing

We want to hash a set $S \subseteq [u]$ with $|S| = n$ using a pairwise independent hash family $H = \{h : [u] \to [m]\}$. Pairwise independence implies that the expected number of collisions is

$$\mathbb{E}\left(\#\text{ collisions}\right) = \binom{n}{2}\mathbb{P}\left(h(x)=h(y)\right) \leq \frac{n^2}{2m}.$$

By choosing $m \geq n^2$ the above becomes less that $1/2$. Thus, we have zero collisions with probability at least $1/2$. This gives us $O(1)$ lookup time, $O(n^2)$ space and $O(1)$ words to store H. We recall that Cuckoo Hashing has $O(1)$ worst case lookup time, $O(n)$ space and $O(n)$ words in order to store H.

Let X_i be the number of items in bin i, then we would like to compute the second moment

$$\mathbb{E}\left(\sum_i X_i^2\right). \tag{1}$$

Since the expected number of collisions is $\mathbb{E}\binom{X_i}{2}$ and we have that $X_i^2 = 2\binom{X_i}{2} + X_i$, we can obtain

$$\mathbb{E}\left(\sum_i X_i^2\right) = 2\mathbb{E}\left(\sum_i \binom{X_i}{2}\right) + \mathbb{E}\left(\sum_i X_i\right) \leq 2\frac{n^2}{m} + n.$$

Which is $O(n)$ for $n = m$. So, we can use the following idea to achieve Perfect Hashing: (i) Pick $h : U \to [n]$ such that $\sum_i X_i^2 \leq 10n$. (ii) For each cell i of the hash table make a secondary hash table with size $\sum_i X_i^2 \leq 10n$ and hash function $h_i : U \to [m_i]$. As we saw before, the probability that there are no collisions in the secondary hash table is small (less than $1/2$) if we choose $m_i \geq X_i^2$.