CSE 521: Design and Analysis of Algorithms I **Spring 2016**

Lecture 4: Universal Hash Functions/Streaming Cont'd

Lecturer: Shayan Oveis Gharan *April 6th* *Scribe: Jacob Schreiber*

4.1 Hash Functions

Suppose we want to maintain a data structure of a set of elements $x_1, \ldots, x_m$ of a universe $\mathcal{U}$, e.g., images, that can perform insertion/deletion/search operations. A simple strategy would be to have one bucket for every possible image, i.e., each element of $\mathcal{U}$, and indicate in each bucket whether or not the corresponding image appeared. Unfortunately, $|\mathcal{U}|$ can be much much larger than the space available in our computers; for example, if $\mathcal{U}$ represents the set of all possible images, $|\mathcal{U}|$ is as big as $2^{1000000}$.

Instead, one may use a hash function. A hash function $h : \mathcal{U} \to [B]$ maps elements of $\mathcal{U}$ to integers in $[B]$. For every element of the sequence we mark $h(x_i)$ with x_i. When a query x arrives, we go to the cell $h(x)$ if no element is stored there, x is not in our sequence. Otherwise, we go over all elements stored in $h(x)$ and see if any of them is equal to x. Observe that the search operation thus depends on the number of elements stored in $h(x)$. Ideally, we would like to have a hash function that stores at most one element in every $0 \leq i \leq B - 1$. Fix a function h. Observe that h maps $1/B$ fraction of all elements of $\mathcal{U}$ to the same number $i \in [B]$. Therefore, the search operation in the worst case is very slow.

We can mitigate this problem by choosing a hash function h uniformly at random the family of all functions that map $\mathcal{U}$ to B; let $\mathcal{H} = h : \mathcal{U} \to [B]$, and let $h \sim \mathcal{H}$ chosen uniformly at random. Now, if the length of the sequence $m \ll B$, then, by the birthday paradox phenomenon, with high probability, no two elements of the sequence map to the same cell. In other words, there is no collisions. However, observe that $\mathcal{H}$ has $|\mathcal{U}|^B$ many functions, so even describing h requires $\log |\mathcal{U}|^B = |\mathcal{U}| \log B$ bits of memory. Recall that we assumed $|\mathcal{U}| \gg 2^{1000000}$ so we cannot efficiently represent h. Instead, we are going to work with smaller much families of functions say $\mathcal{H}^*$; such a family can only guarantee weaker notions of independence, but because $|\mathcal{H}^*| \ll |\mathcal{H}|$, it is much easier to describe a randomly chosen function from $\mathcal{H}^*$.

4.2 2-Universal Functions

In this section, we describe a family hash functions that only preserve pairwise-independent. Let p be a prime number, and let $\mathcal{H} = \{h : [p] \to [p], h(x) = ax + b \mod p\}$. Observe that any function $h_{a,b} \in \mathcal{H}$ can be represented in $O(\log p)$ bits of memory just by recording the $a, b \in [p]$. Next, we show that a uniformly random function $h \sim \mathcal{H}$ is pairwise independent.

Lemma 4.1. *For any $x, y, c, d \in [p] x \neq y, \mathbb{P}[h(x) = c, h(y) = d] = \frac{1}{p^2}$*

Proof. Suppose for some $x \neq y$,

$$h(x) \equiv c, \text{ and } h(y) \equiv d.$$

Equivalently, we can write,

$$ax + b \equiv c \mod p, \text{ and } ay + b \equiv d \mod p.$$

Using the laws of modular equations, we can write,

$$a(x-y) \equiv (c-b) - (d-b) \mod p.$$

Since p is a prime, any number $1 \le z \le p-1$ has a multiplicative inverse, i.e., there is a number $1 \le z^{-1} \le p-1$ such that $p \cdot p^{-1} \equiv 1 \mod p$. Since $x \neq y$, $x - y \neq 0$. Therefore, it has a multiplicative inverse, and we can write,

$$a = (x-y)^{-1}(c-d) \mod p,$$

which gives,

$$b = d - ay \mod p.$$

In words, having x, y, c, d uniquely defines a, b. Since there are p^2 possibilities for a, b, we get

$$\mathbb{P}[h(x) = c, h(y) = d] = 1/p^2.$$

□

For our applications in estimating F_0, we first need to choose a prime number $p > n$. Then, we can use a hash function $h : [n] \to [B]$ where for any $0 \le x \le n-1$, $h(x) = ax + b \mod p \mod B$. It is easy to see that such a function is almost pairwise independent which is good enough for our application in estimating F_0.

We can extend the above construction to a family of k-wise independence hash functions. We say a hash function $h : [p] \to [p]$ is k-wise independent if for all distinct $x_0, \ldots, x_{k-1}$,

$$\mathbb{P}[\forall i, h(x_i) = c_i] = \frac{1}{p^k}.$$

Such a hash function h can be constructed by choosing $a_0, a_1, \ldots, a_{k-1}$ uniformly and independently from $[p]$ and letting

$$h(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} ... a_1 x + a_0.$$

We are not proving that this will give a k-wise independence hash function. Instead, we just give the high-level idea. Let h be a 4-wise independent hash function and let $x_0, x_1, x_2, x_3 \in [p]$ be distinct and $c_0, c_1, c_2, c_3 \in [p]$ we need to show that there is a unique set a_0, a_1, a_2, a_3 for which $h(x_i) = c_i$ for all i. To find a_0, a_1, a_2, a_3 it is enough to solve the following system of linear equautions.

$$\begin{bmatrix} x_0^3 & x_0^2 & x_0 & 1 \\ x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ x_3^3 & x_3^2 & x_3 & 1 \end{bmatrix} \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

It turns out that the Matrix in the LHS has a nonzero determinant of x_0, x_1, x_2, x_3 are distinct. In such a case, it is invertible, and we can use the inverse to uniquely define a_0, a_1, a_2, a_3.

4.3 F_2 Moment

Before designing a streaming algorithm that estimates F_2, let us revisit the random walk example that we had a few lectures ago. Let $X = \sum_i X_i$ where for each i,

$$X_i = \begin{cases} +1, & \text{w.p. } \frac{1}{2} \\ -1, & \text{w.p. } \frac{1}{2} \end{cases}$$

Using the Hoeffding bound, we previously showed that for any $c > 2$, $\mathbb{P}\left[X \le c\sqrt{n}\right] \ge 1 - e^{\frac{-c^2}{2}}$. Is this bound tight? Can we show that $X \ge \Omega(n)$ with a constant probability? The answer yes. More generally it follows from the central limit theorem. But instead of using such a heavy tool there is a more elementary argument that we can use. To show that $X \ge \Omega(\sqrt{n})$ with a constant probability, it is enough to show that $\mathbb{E}\left[X^2\right] \ge n$.

$$\begin{aligned}\mathbb{E}\left[X^2\right] = \mathbb{E}\left[\sum_i X_i]^2\right] &= \mathbb{E}\left[\sum_{i,j} X_i X_j\right] \\ &= \sum_{i,j} \mathbb{E}\left[X_i X_j\right] = \sum_i \mathbb{E}\left[X_i^2\right] = n,\end{aligned}$$

where in the second to last equality we use that X_i, X_j are independent, so $\mathbb{E}\left[X_i X_j\right] \neq 0$ only when $i = j$, and in the last equality we use $\mathbb{E}\left[X_i^2\right]$ is 1.

Now back to estimating F_2. We want to use a similar idea. Let $x_1, x_2, \dots, x_m \in [n]$ be the input sequence. For each $i \in [n]$ let $m_i := \#\{x_j = i\}$. Recall that

$$F_2 := \sum_{i=1}^{n} m_i^2.$$

Let $h : [n] \to \{+1, -1\}$ where for any $i \in [n]$,

$$h(i) = \begin{cases} +1, & \frac{1}{2} \\ -1, & \frac{1}{2}, \end{cases}$$

chosen independently. Consider the following algorithm: Start with $Y = 0$. After reading each x_i, let $Y = Y + h(x_i)$. Return Y^2.

Before, analyzing the algorithm let us study two extreme cases. First assume that $x_1 = x_2 = \cdots = x_m$. Then, $Y = m$, $Y^2 = m^2$ as desired. Now, assume that $x_1, x_2, dots, x_m$ are mutually distinct, then the distribution of Y is the same as a random walk of length m; so by the previous observation $Y \approx \sqrt{n}$ and $Y^2 \approx n$ as desired.

Lemma 4.2. *Y^2 is an unbiased estimator of F_2, i.e., $\mathbb{E}\left[Y^2\right] = F_2$.*

Proof. First, observe that

$$Y = \sum_i m_i h(i).$$

Therefore,

$$\begin{aligned}\mathbb{E}\left[Y^2\right] = \mathbb{E}\left[\sum_{i,j} m_i m_j h(i) h(j)\right] &= \sum_{i,j} m_i m_j \mathbb{E}\left[h(i)h(j)\right] \\ &= \sum_i m_i^2 \mathbb{E}\left[h(i)^2\right] = \sum_i m_i^2\end{aligned}$$

where the second to last equality uses that $h(i)$ is independent of $h(j)$ for all $i \neq j$. □

Now, all we need to do is to estimate the expectation of Y^2 within a $1 \pm \epsilon$ factor. By Chebyshev's inequality all we need to show is that Y^2 has a small variance.

Lemma 4.3. $\mathrm{Var}(Y^2) \leq 2\mathbb{E}\left[Y^2\right]^2$.

Proof. First, we calculate $\mathbb{E}\left[Y^4\right]$. The idea is similar to before, we just use the independence of $h(i)$'s.

$$\begin{aligned}\mathbb{E}\left[Y^4\right] &= \mathbb{E}\left[\sum_{i,j,k,l} m_i m_j m_k m_l h(i)h(j)h(k)h(l)\right] \\ &= \sum_{i,j,k,l} m_i m_j m_k m_l \mathbb{E}\left[h(i)h(j)h(k)h(l)\right] = \sum_i m_i^4 \mathbb{E}\left[h(i)^4\right] + 6\sum_{i<j} m_i^2 m_j^2 \mathbb{E}\left[h(i)^2 h(j)^2\right]\end{aligned}$$

To see the last equality, observe that for any 4-tuple, i, j, k, l, $\mathbb{E}\left[h(i)h(j)h(k)h(l)\right]$ is nonzero only if each integer in $[m]$ shows up an even number. In other words, there are only two cases where $\mathbb{E}\left[h(i)h(j)h(k)h(l)\right]$ is nonzero: (i) when $i = j = k = l$, (ii) when two of these four numbers are equal and the other two are also equal.

Since for each i, $\mathbb{E}\left[h(i)^2\right] = \mathbb{E}\left[h(i)^4\right] = 1$, we have

$$\mathbb{E}\left[Y^4\right] = \sum_{i=1}^{n} m_i^4 + 6\sum_{i<j} m_i^2 m_j^2.$$

Now, using Lemma 4.2, we can write,

$$\mathrm{Var}(Y^2) = \mathbb{E}\left[Y^4\right] - \mathbb{E}\left[Y^2\right]^2 = 4\sum_{i<j} m_i^2 m_j^2 \leq 2\mathbb{E}\left[Y^2\right]^2$$

as desired. □

Now, all we need to do is to use independent samples of Y^2 to reduce the variance. Suppose we take k independent samples of Y^2 using k independently chosen hash functions $h_1, \ldots, h_k$, i.e., we run the following algorithm: Start with $Y_1 = Y_2 = \cdots = Y_k = 0$. After reading x_i, let $Y_j = Y_j + h(x_i)$ for all $1 \leq j \leq k$. Then,

$$\mathrm{Var}\left(\frac{1}{k}(Y_1^2 + \cdots + Y_k^2)\right) = \frac{1}{k}\mathrm{Var}(Y^2).$$

Therefore, by the Chebyshev's inequality, we can write,

$$\begin{aligned}\mathbb{P}\left[\left|\frac{1}{k}\sum_i Y_i^2 - \mathbb{E}\left[Y^2\right]\right| \geq \epsilon\mathbb{E}\left[Y^2\right]\right] &\leq \frac{\mathrm{Var}(\frac{1}{k}\sum_{i=1}^{k} Y_i^2)}{\epsilon^2\mathbb{E}\left[Y^2\right]^2} \\ &= \frac{\frac{1}{k}2\mathbb{E}\left[Y^2\right]^2}{\epsilon^2\mathbb{E}\left[Y^2\right]^2} = \frac{2\epsilon^2}{k}\end{aligned}$$

So, $k = \frac{5}{\epsilon^2}$ many samples is enough to approximate F_2 within $1+\epsilon$ factor with probability at least $\frac{9}{10}$. Note that in the above construction we assumed that $h(.)$ assigns independent values to all integers in $[n]$. But, it can be seen from the proof that we only used 4-wise independence. The only place that we used independence was to show that $\mathbb{E}\left[h(i)h(j)h(k)h(l)\right] = 0$ when i, j, k, l are mutually distinct. That is of course true even if $h(.)$ is just a 4-wise independent function. Taking that into account we can run the above algorithm with space $O(\log(n)/\epsilon^2)$.

In addition, we can turn the above probabilistic guarantee into $1-\delta$ probability using $\frac{\log\frac{1}{\delta}}{\epsilon^2}$ many samples. We refrain from giving the details. For more detailed discussion we refer to [AMS96].

References

[AMS96] N. Alon, Y. Matias, and M. Szegedy. “The space complexity of approximating the frequency moments”. In: *STOCw*. ACM. 1996, pp. 20–29 (cit. on p. 4-4).