CSE 521: Design and Analysis of Algorithms I **Fall 2018**

Lecture 4: Hashing and Streaming Algorithms

Lecturer: Shayan Oveis Gharan *10/10/2018* *Scribe:*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

At the end of the previous lecture, we introduced the basic notions of hashing and saw some of its applications. In this lecture, we are going to study hashing in more detail.

4.1 The Problem of Hashing

Let $U = \{0, 2, ..., 2^{100000} - 1\}$ be a large universe of numbers, let $X_1, ..., X_n \in U$ be n input numbers in such a universe where $n \ll |U|$. We think of them as images. Recall from the last lecture, our goal is to construct a family of hash functions $\mathcal{H}$, where every function in this family maps from U to $[N] := \{0, 2, ..., N-1\}$. We want to store these images in a data structure in order to be able to answer the following query in constant time, $O(1)$: Given an image Y, is there an image $X_i = Y$?

Throughout this document we always write $[p]$, for an integer $p > 0$ to denote the set $\{0, 1, \dots, p-1\}$.

Recall that in the example of birthday paradox, where we have n people and N days in a year, the probability that two people were born in the same day is small when $N \gg n^2$. Therefore, if we map the n input numbers uniformly to $\{1, 2, ..., cn^2\}$ for some large enough constant c, then by an analysis similar to what we did in birthday paradox, we can show that the probability of collision is small. However, the problem here is that to record the uniform hash function, we need $|U| \log N$ bits, which is too big. Therefore, instead of choosing uniformly from all the possible mappings, we choose uniformly from a smaller set of functions. This motivates us to use hash functions with limited independence.

4.2 Limited Independence

Definition 4.1 (One-way Independence)**.** *Let $\mathcal{H}$ be a family of hash functions, we say $\mathcal{H}$ is one-way independent if for all $X_1 \in U$ and for all $a_1 \in [N]$, we have*

$$\mathbb{P}_{h \sim H}[h(x_1) = a_1] = \frac{1}{N}.$$

Note that the above definition of one-way independence is not enough for a good family of hash functions. The family of constant functions $\{h_1, h_2, ..., h_N\}$ where $h_i(x) = i$ for every $x \in U$ is one-way independent. Constant functions give us the largest amount of collisions we can imagine. However, when we take one step further and use a pairwise independent family of functions, we are able to achieve small collision probability.

Definition 4.2 (Pairwise Independence)**.** *Let $\mathcal{H}$ be a family of hash functions, we say $\mathcal{H}$ is pairwise independent if for all distinct $x_1, x_2 \in U$ and for all $a_1, a_2 \in [N]$, we have*

$$\mathbb{P}_{h \sim H}[h(x_1) = a_1, h(x_2) = a_2] = \frac{1}{N^2}.$$

In fact, in the case of our problem, hash function families with the below definition of approximate pairwise independence property is sufficient.

Definition 4.3 (Approximate Pairwise Independence)**.** *Let $\mathcal{H}$ be a family of hash functions, we say $\mathcal{H}$ is α-approximate pairwise independent if for all distinct $x_1, x_2 \in U$ and for all $a_1, a_2 \in [N]$, we have*

$$\underset{h\sim H}{\mathbb{P}}[h(x_1) = a_1, h(x_2) = a_2] \leq \frac{\alpha}{N^2}.$$

Before proving that pairwise independence is sufficient for a good family of hash functions, we remark that we can extend the definitions 4.1 and 4.2 to k-wise independence. Although pairwise independence is already sufficient for our application today, k-wise independent hash functions are very important objects in computer science, and thus have found a lot of applications elsewhere.

Definition 4.4 (k-wise Independence)**.** *Let $\mathcal{H}$ be a family of hash functions, we say $\mathcal{H}$ is k-wise independent if for all distinct $x_1, x_2, ...x_k \in U$ and for all $a_1, a_2, ..., a_k \in [N]$, we have*

$$\underset{h\sim H}{\mathbb{P}}[h(x_1) = a_1, h(x_2) = a_2, ..., h(x_k) = a_k] = \frac{1}{N^k}.$$

4.3 Birthday Paradox Revisit

Suppose we $\mathcal{H} = \{h : U \to [N]\}$ is a pairwise independent family of hash functions. Given n images $X_1, \ldots, X_n$, we choose a function $h \sim \mathcal{H}$ uniformly at random and we map each X_i to $h[X_i]$. We prove the following lemma:

Lemma 4.5. *If $N \geq \alpha n^2$ then with probability $1/2$ there is no collision.*

Note that assuming this lemma we can simply choose multiple $h \sim \mathcal{H}$ until we get no collision.

Now, let us revisit the analysis of the birthday paradox. We will see that the actual property that we need is approximate pairwise independence instead of mutual independence.

Similar to the analysis of the birthday paradox, we define Y_{ij} to be the indicator random variables that $h(X_i) = h(X_j)$, and let $Y = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Y_{ij}$. Suppose that $\mathcal{H}$ is an α-approximate pairwise independent hash function family, then for every distinct $i, j \in [n]$

$$\mathbb{E}[Y_{ij}] = \underset{h\sim\mathcal{H}}{\mathbb{P}}[h(X_i) = h(X_j)] = \sum_{a\in[N]} \underset{h\sim\mathcal{H}}{\mathbb{P}}[h(X_i) = a, h(X_j) = a] \leq \frac{\alpha}{N^2} \cdot N = \frac{\alpha}{N}.$$

Therefore

$$\mathbb{E}[Y] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[Y_{ij}] \leq \binom{n}{2} \alpha/N.$$

So if $\alpha n^2 < N$, then by Markov's inequality, we have

$$\underset{h\sim\mathcal{H}}{\mathbb{P}}[\forall \text{ distinct } i, j \in \{1, ..., n\}, h(X_i) \neq h(X_j)] = \mathbb{P}[Y = 0] \geq \frac{1}{2}$$

The above analysis shows that a family of hash functions with the property of α-approximate pair independence for some constant α would be suffice for our purpose.

As we will see in the following sections, to store the (approximate) pairwise independence hash functions, we will need $O(\log |U|)$ space. Therefore, the main downside of the above method is a quadratic loss in memory, i.e., to store n images we need to use $O(n^2)$ memory.

4.4 Double Hashing

The material of this section follows from the work of Fredman et al. [FKS84]. In this section we see how to use a two layers hashing scheme to reduce the memory size to $O(n)$.

Instead of choosing $N = \Theta(n^2)$, we choose $N = n$ and we first choose $h \sim \mathcal{H}$ to map all images $X_1, \dots, X_n$ to to N buckets. Note that for this choice of N we will have many collisions. Say Z_i be the random variable is the number of images that map to the i-th location. We choose another family pairwise independent hash functions $\mathcal{H}_i$ which map $U \to [N_i]$ for $N_i = \alpha Z_i^2$. Then, we choose $h \sim \mathcal{H}_i$ for a second layer of hashing; we choose $h_i \sim \mathcal{H}_i$. Then, we map each of the Z_i images which map to location i by h to one of $[N_i]$ locations using h_i. By the analysis of the previous section the probability of collision in the second layer is at most $1/2$. So, we can test multiple samples for h_i until we get one with no collisions.

This double hashing method has obviously an $O(1)$ search time. Given a query Y, first we find $h(Y)$; say $h(Y) = i$. Then, we compute $h_i(Y)$ for the second layer. If no image is stored in $h_i(Y)$ we output "no". Otherwise, we check Y agains the unique image stored in $h_i(Y)$ and we output "yes" if they are the same and "no" otherwise.

Now, let us compute the expected size of the memory of this double hashing scheme. We use $O(n \log |U|)$ to store $n+1$ hash functions. The expected number of memory locations is at most

$$\mathbb{E} \sum_{i=1}^{N} \alpha Z_i^2 = \alpha \left(2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E} Y_{ij} + n \right) = \frac{2\alpha^2 \binom{n}{2}}{N} + \alpha n = O(\alpha^2 n).$$

The see the first identity note that

$$Z_i = \sum_{k=1}^{n} \mathbb{I}\left[h(X_k) = i\right].$$

Therefore,

$$Z_i^2 = Z_i + 2 \sum_{1 \le k < \ell \le n} \mathbb{I}\left[h(X_k) = h(X_\ell) = i\right]$$

So,

$$\sum_{i=1}^{n} Z_i^2 = \sum_{i=1}^{n} Z_i + 2 \sum_{1 \le k < \ell \le n} \mathbb{I}\left[h(X_k) = h(X_\ell)\right] = \sum_{i=1}^{n} Z_i + 2 \sum_{1 \le k < \ell \le n} Y_{k,\ell}.$$

Taking expectation from both sides proves the identity.

4.5 Construction of Pairwise Independent Hash Function

Let p be a prime so that $|U| \le p \le 2|U|$. Let variables a and b both be uniformly chosen from $\{0, ..., p-1\}$. We show that the family of functions $f_{a,b}(x) = ax + b \mod p$ is pairwise independent. Note that these functions map $[p] \to [p]$. Later we see that using a mod operation we can construct pairwise independent hash functions that map $[p] \to [N]$.

Claim 4.6. *For all $x, y \in \{0, ..., p-1\}$ such that $x \neq y$, and for all $s, t \in \{0, ..., p-1\}$, we have*

$$\mathbb{P}_{a,b}[f_{a,b}(x) = s, f_{a,b}(y) = t] = \frac{1}{p^2}$$

Proof. When $f_{a,b}(x) = s$ and $f_{a,b}(y) = t$, we have

$$\begin{aligned} ax + b &\equiv s \mod p \\ ay + b &\equiv t \mod p. \end{aligned}$$

Subtracting one from the other, we get

$$a(x-y) \equiv s - t \mod p.$$

Since p is a prime number, for every number $k \in \{1, ..., p-1\}$, there is a unique (modular) inverse k^{-1} of k so that $kk^{-1} \equiv 1 \mod p$. We do not discuss the algorithm for finding modular inverse, we refer students to https://en.wikipedia.org/wiki/Modular_multiplicative_inverse for details.

Since $x \neq y$, $x - y$ has a modular inverse. So, we can write solve the above system of modular equations for a and b; in particular, we have

$$a \equiv (s-t)(x-y)^{-1} \mod p.$$

Furthermore,

$$b \equiv s - ax \mod p$$

The above analysis shows that for a fixed x, y the following holds: Given any $s, t \in [p]$ there exists a pair (a, b) so that $f_{a,b}(x) = s$ and $f_{a,b}(y) = t$. Since there are p^2 possible options for (a, b) to take and p^2 many options for (s, t) this mapping is one-to-one. Therefore,

$$\mathbb{P}_{a,b}[f_{a,b}(x) = s, f_{a,b}(y) = t] = \frac{1}{p^2}$$

as desired. □

Now, we choose the family of hash functions to be $\mathcal{H} = \{h_{a,b}\}$ where

$$h_{a,b}(x) = f_{a,b}(x) \mod N.$$

Note that to store this function in memory, we only have to store a and b, which takes only $O(\log p) = O(\log |U|)$ many bits. For the particular application to Hashing universe U, we can use another idea to reduce the memory size to $O(\log n)$. Please refere to [FKS84, Lem 2] for details.

Now, we show that $\mathcal{H}$ is the family of hash functions with 2-approximate pairwise independence property.

Claim 4.7. *For all $x, y \in U$ so that $x \neq y$, we have*

$$\mathbb{P}_{a,b}[h_{a,b}(x) = h_{a,b}(y)] \leq \frac{1}{N} + \frac{1}{p}.$$

Proof. For all $x, y \in U$ so that $x \neq y$, $h_{a,b}(x) = h_{a,b}(y)$ if and only if $f_{a,b}(x) \equiv f_{a,b}(y) \mod N$. Thus, by Claim 4.6

$$\begin{aligned} \mathbb{P}_{a,b}[h_{a,b}(x) = h_{a,b}(y)] &= \mathbb{P}_{a,b}[f_{a,b}(x) \equiv f_{a,b}(y) \mod N] \\ &= \sum_{0 \leq s,t < p : s \neq t} \mathbb{P}\left[f_{a,b}(x) = s, f_{a,b}(y) = t\right] \mathbb{I}\left[s \equiv t \bmod N\right] \\ &= \sum_{0 \leq s,t < p : s \neq t} \frac{\mathbb{I}\left[s \equiv t \bmod N\right]}{p^2} \\ &\leq \frac{p \left\lceil \frac{p}{N} \right\rceil}{p^2} \leq \frac{1}{N} + \frac{1}{p} \end{aligned}$$

The first inequality follows by the fact that for any $s \in [p]$ there are at most $\lceil p/n \rceil$ numbers t such that $s \equiv t \bmod N$. □

Note that the family of functions that we construct above is approximately pairwise independent but that is enough for all interesting applications.

We remark that we can extend the above construction and obtain a family of hash functions that is k-wise independent. For some prime number p, consider the family of hash functions

$$f_{a_0,\ldots,a_{k-1}}(x) = a_{k-1}x^{k-1} + \ldots + a_1 x + a_0,$$

where $a_0, \ldots, a_{k-1}$ are uniformly chosen in $\{0, 1, \ldots, p-1\}$. Similar to the invertible argument we used above, the proof that this construction is k-wise independent follows from the fact that the Vandermonde matrix

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_k \\ x_1^2 & x_2^2 & \cdots & x_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{k-1} & x_2^{k-1} & \cdots & x_k^{k-1} \end{bmatrix}$$

is invertible for distinct $x_1, \ldots, x_k$. So, in general we can store a k-wise independent hash function with only $O(k \log |U|)$ amount of memory.

References

[AMS96] N. Alon, Y. Matias, and M. Szegedy. “The space complexity of approximating the frequency moments”. In: *STOC*. ACM. 1996, pp. 20–29 (cit. on p. 4-5).

[FKS84] M. L. Fredman, J. Komlós, and E. Szemerédi. “Storing a Sparse Table with 0(1) Worst Case Access Time”. In: *J. ACM* 31.3 (June 1984), pp. 538–544 (cit. on pp. 4-3, 4-4).