CS 388R: Randomized Algorithms, Fall 2019 September 26th, 2019

Lecture 9: Limited Independence

Prof. Eric Price *Scribe: Yangxinyu Xie, Sabee Grewal*

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

1 Overview

We begin by listing the upsides and downsides of fully independent families of hash functions and compare them with the case of limited independence.

Full Independence

+ Let's you analyze things really well because you know exactly what's happening. You can do tight analysis (with means, variances, concentrations (cf. Chernoff bounds)).
+ Assuming full independence is fairly accurate even if your hash function is not. For example, non-adversarial input (for example, high entropy inputs) and cryptographic functions are examples where assuming full independence is justified.
− You cannot easily store/lookup your hash values. Of course, depending on the situation you may not care. Take load balancing, for example. You're given a task that needs to be assigned to some machine. You can randomly your machine, assign the task, and your job is complete. Whereas with a hash table, you must be able to lookup your item once it's stored.

Limited Independence

+ You can get guarantees for all inputs, even if they're adversarial.
+ You can store the hash function.
+ You can typically still analyze means, variances, and $O(1)$-moments.
− You cannot compute $O(\log n)$-moments (which means you cannot use Chernoff bounds.)

2 Introduction and Definitions

Definition 1. *A family of hash functions $\mathcal{H}$ is called universal if $h_i : \mathcal{U} \to [m]$ such that $\forall x, y \in \mathcal{U}, x \neq y, a, b \in [m]$*

$$\mathbb{P}[h(x) = h(y)] \leq \frac{1}{m} \tag{1}$$

Definition 2. *A family of hash functions $\mathcal{H}$ is called approximately universal if $h_i : \mathcal{U} \to [m]$ such that $\forall x, y \in \mathcal{U}, x \neq y, a, b \in [m]$ and some $\epsilon > 0$,*

$$\mathbb{P}[h(x) = h(y)] \leq \frac{1+\epsilon}{m} \tag{2}$$

Definition 3. *A family of hash functions $\mathcal{H}$ is called pairwise independent if $h_i : \mathcal{U} \to [m]$ such that $\forall x, y \in \mathcal{U}, x \neq y, a, b \in [m]$*

$$\mathbb{P}[h(x) = a, h(y) = b] \leq \frac{1}{m^2} \tag{3}$$

Definition 4. *A family of hash functions $\mathcal{H}$ is called approximately pairwise independent if $h_i : \mathcal{U} \to [m]$ such that $\forall x, y \in \mathcal{U}, x \neq y, a, b \in [m]$ and some $\epsilon > 0$,*

$$\mathbb{P}[h(x) = a, h(y) = b] \leq \frac{1+\epsilon}{m^2} \tag{4}$$

3 Standard Hash Table

The key claim to fame for standard hash tables is expected constant time query while only using linear space. In this section, we'll discuss space requirements of hash tables and show that pairwise independence (or, in fact, universality) suffices for constant time queries.

Let $\mathcal{U}$ be a universe of items and S be the subset of words that we're trying to store. We're going to map these items under a function $h \in \mathcal{H}$ to a table of size m. That is, for each item in S, our hash function maps it to some index from 1 to m. We'll handle collisions by chaining items together with linked lists.

Space complexity. A single word takes $\log|\mathcal{U}|$ bits to store and we are trying to store $|S|$ words. Additionally, we need $\log|\mathcal{H}|$ bits to store our hash function. So, in total we need $O(|S|\log|\mathcal{U}| + \log|\mathcal{H}|)$ space. If we can make our hash function sufficiently small, then our space complexity is linear in the number of words we're trying to store (i.e. we require $O(|S|)$ space). In the next section we'll see that we can get a small hash family (and therefore be able to construct a standard hash table with linear space) when we have pairwise independence.

Time complexity. How long does it take to lookup an $x \in \mathcal{U}$? Roughly speaking, the total query time scales with the length of the linked list at a given cell (i.e. the number of elements $y \in S \mid h(x) = h(y)$) plus time it takes to compute $h(x)$. More formally, let T be the time it takes to lookup an $x \in \mathcal{U}$ and let $T_{h(x)}$ be the time to compute $h(x)$. We're interested in the expectation of T over some random hash function from some family $\mathcal{H}$. We can solve for $\underset{h\in\mathcal{H}}{\mathbb{E}}[T]$ like so:

$$\begin{aligned}
\underset{h\in\mathcal{H}}{\mathbb{E}}[T] &\le T_{h(x)} + \sum_{y\in S} \underset{h\in\mathcal{H}}{\mathbb{P}}[h(x)=h(y)] \\
\underset{h\in\mathcal{H}}{\mathbb{E}}[T] &\le T_{h(x)} + \sum_{y\in S} \frac{2}{m} \\
\underset{h\in\mathcal{H}}{\mathbb{E}}[T] &\le T_{h(x)} + 1 + \frac{2}{m}|S| \\
\underset{h\in\mathcal{H}}{\mathbb{E}}[T] &\le T_{h(x)} + O(1) \\
\underset{h\in\mathcal{H}}{\mathbb{E}}[T] &\le O(1).
\end{aligned}$$

The first step follows from the fact that h is from a universal family of hash functions (see Definitions 1 and 2). In our case, we use Definition 2 with $\epsilon = 1$. The last step follows from the assumption that computing our hash function takes constant time.

Our analysis shows that queries to our table complete in constant time in expectation. However, we rely on the fact that our hash functions are pairwise independent and can be stored efficiently. In the next section, we discuss how this is accomplished.

4 Pairwise Independent Hash Functions

Let $M = 2^m$ and $U = 2^u$, we fill a matrix $A \in \mathbb{R}^{m\times u}$ with random bits and then pick a random vector $b \in \{0,1\}^m$. Define a hash family $\mathcal{H}$ in field $\mathbb{F}_2$

$$\mathcal{H} = \{h_{A,b}(x) = Ax + b\} \tag{5}$$

The space of used by the hash function is the space to store A and b, which is $O(um) + O(u) = O(um)$. The time it takes to compute is again $O(um)$.

Theorem 5. *$\mathcal{H}$ is pairwise independent.*

Proof. Given $x, y \in \{0,1\}^u$, suppose $h_{A,b}(x) = \alpha, h_{A,b}(y) = \beta, x \ne y$ where $\alpha, \beta \in \{0,1\}^m$. Let $z = x - y$ and $\gamma = \alpha - \beta$, we have $z \ne \{0\}^m$ since $x \ne y$ and $Az = \gamma$. Therefore, we have

$$\begin{aligned}
\mathbb{P}[A_1 z = \gamma_1] &= 1/2 \\
\mathbb{P}[A_2 z = \gamma_2 | A_1] &= 1/2 \\
&\vdots \\
\mathbb{P}[A_m z = \gamma_m | A_{m-1}, \ldots, A_1] &= 1/2
\end{aligned}$$

Thus,

$$\mathbb{P}[Az = \gamma] = \mathbb{P}[A_m z = \gamma_m | A_{m-1}, \ldots, A_1] \ldots \mathbb{P}[A_2 z = \gamma_2 | A_1]\mathbb{P}[A_1 z = \gamma_1] = (1/2)^m \tag{6}$$

Given A, we have $b = \alpha - Ax$. Since b and A are chosen independently, we have $\mathbb{P}[b = \alpha - Ax] = (1/2)^m$. Overall,

$$\mathbb{P}[h_{A,b}(x) = \alpha, h_{A,b}(y) = \beta] = (1/2)^m (1/2)^m = (1/2)^{2m} \tag{7}$$

□

Corollary 6. *$\mathcal{H}$ is universal.*

Proof. By theorem 4.1, we have

$$\mathbb{P}[h_{A,b}(x) = \alpha, h_{A,b}(y) = \beta] = (1/2)^{2m} \tag{8}$$

Let $\alpha = \beta$, we then have 2^m choices of $\alpha \in \{0,1\}^m$, then

$$\mathbb{P}[h_{A,b}(x) = h_{A,b}(y)] = (1/2)^{2m} 2^m = (1/2)^m \tag{9}$$

□

4.1 Random Toeplitz Matrix

An improvement of the above construction is to use Toeplitz Matrix. We construct a matrix $A \in \mathbb{R}^{m \times u}$ by populating the first row and first column of A with random $\{0,1\}$ entries and letting any other entry $A_{ij} = A_{i-1,j-1}$. Theorem still holds true in this construction. To see this, we have

$$A_{2,1}z_1 = \gamma_2 - (A_{2,2}z_2 + ... + A_{2,m}z_m) = \gamma_2 - (A_{1,1}z_2 + ... + A_{2,m-1}z_m)$$
$$\vdots$$

Since the first column is picked at random, we have again

$$\mathbb{P}[A_2 z = \gamma_2 | A_1] = 1/2$$
$$\vdots$$
$$\mathbb{P}[A_m z = \gamma_m | A_{m-1}, \dots, A_1] = 1/2$$

The space of used by the hash function is the space to store the first row and the first column of A as well as b, which is $O(u+m)$. The time it takes to compute is till $O(um)$.

4.2 A 2-approximate Universal Hashing

Pick a random odd number a in $[M]$. We define a hash family $\mathcal{H}$

$$\mathcal{H} = \{h_a(x) = (ax \mod U)/2^{u-m}\} \tag{10}$$

The following theorem can be shown, we direct the reader to [DHKP97] for the proof.

Theorem 7.

$$\mathbb{P}[h(x) = h(y)] \leq 2/M \tag{11}$$

5 k-wise Independent Hash Functions

Again, let $M = 2^m$ and $U = 2^u$, consider the hash family

$$\mathcal{H} = \{h_a(x) = (ax + b) \mod M\} \tag{12}$$

where a, b are chosen independently.

5.1 $M = U$

It is easy to see that this hash family is pairwise independent. For $x, y \in [U], x \neq y$

$$\mathbb{P}[ax + b = \alpha, a'x + b' = \beta] = (\frac{1}{M})^2 = (\frac{1}{U})^2 \tag{13}$$

The time to calculate the hash function is $O(1)$ and the space to store the hash function is also $O(1)$.

5.1.1 $M < U$

As M is less than U, we have the hash family is $(1 + M/U)$ approximately pairwise.

$$\mathbb{P}[ax + b = \alpha, a'x + b' = \beta] = (\frac{|S|}{U})^2 = (\frac{1}{M})^2(1 \pm \frac{M}{U})^2 \tag{14}$$

where $\lfloor \frac{U}{M} \rfloor \leq S \leq \lceil \frac{U}{M} \rceil$.

5.2 k-wise independent hash function

Consider instead a generalised hash family

$$\mathcal{H} = \{h_a(x) = \sum_{i=0}^{k-1} a_i x^i \mod M\} \tag{15}$$

where a_i is independently generated from $[M]$ for all i.

The proof of k-wise independence is similar to above.

References

[CW79] Lawrence J Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979.

[DHKP97] Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25(1):19–51, 1997.