University of Oulu

Faculty of Information Technology and Electrical Engineering

Randomized Algorithms

521149S

Hash Functions based on Randomized Algorithms

Tri Hong Nguyen

March 13, 2019

Abstract

The hash function is considered as a perfect solution for several applications including error-correcting codes, randomization function, lossy compression, fingerprints and checksums. Due to the important utilization of the hash function, this assignment discusses the hash function and approaches to construct good hash functions. Particularly, the main idea of a good hash function is utilized the randomized algorithm to form a term called universal hashing. From the design of universal hashing, other approaches to design the good hash function are developed such as perfect hashing and Count-Min hashing.

1 Introduction

Hash functions followed by [1] are functions that take inputs with any length to form fixed length outputs. This idea of hash function has been adopted in cryptography topics to become *cryptographic hash functions*. This name is then widely used in cryptography as hashcode, hash total, hash result etc. [1] also classified the hash function into two main categories such as Message Authentication Code (MAC) and Manipulation Detection Code (MDC). Moreover, MDC class is separated into One-Way Hash Functions (OWHF) and Collision Resistant Hash Functions (CRHF) as Fig.1. The term CRHF is viewed as a preferable hash function with an explanation of its property in which the collision-free hash function is discussed in the Section 2.2. In another type of hash function, the first OWHF is proposed by [2] with the emphasis on the difference between weak and strong one-way hash functions. After that, the idea of universal hash function [3] is proposed through the random choice of hash functions from a large set.

The assignment is to focus on the hash functions from its basic ideas until the utilization of randomized algorithms. In more details, the universal hashing is mentioned before the extension to perfect hashing and also Count-Min sketch. These approaches are discussed through their probability of collisions with their required parameters.

The remaining of the assignment is structured as follows: after Section 2 about general hash functions, the universal class of hash functions is mentioned in Section 3. The assignment then shows the idea of perfect hashing which is based on two levels of utilization of universal hashing through Section 4. Finally, another approach based on the idea of the universal hash function (count-min sketch) is described in Section 5. Finally, the summary of the assignment is mentioned in Section 6.

2 Hash Functions

This section displays a general background of hash functions through its requirements. Firstly, several conditions of one-way hash functions and collision

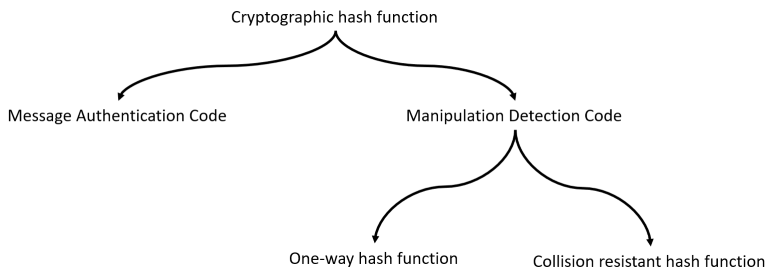
Cryptographic hash function
Message Authentication Code
Manipulation Detection Code
One-way hash function
Collision resistant hash function

Figure 1: The classification of hash function [1]

resistant hash functions are discussed. Next, the collision-free hash function and what are good hash functions? are mentioned.

2.1 One-way and Collision resistant hash function

Following [1], an analysis of two hash function's types is described in this subsection. First of all, a one-way hash function is required to satisfy conditions as

- *No secret description* - the public description of the hash function h is required.
- *Arbitrary length of inputs and fixed length of outputs* - the input' size for hash function h is arbitrary, while that of output is fixed as the number of bits (greater or equal to 64 bits).
- *Low computational cost* - given an input x, the computation cost of $h(x)$ should be cheap or eazy.
- *One-way function* - there are two requirements of this condition. Firstly, with a given output $h(x) = y$, it becomes too really difficult to find back the input x. In addition, given two values x and $h(x)$, it is also difficult to find another x' with $x' \neq x$ and $h(x') = h(x)$.

In term of collision resistant hash function, there are also some similar conditions to one-way hash function; however, it requires an additional requirement (the last one).

- *No secret description* - the hash function's information is public without any secret.
- *Arbitrary length of inputs and fixed length of outputs* - the input' size of the collision hash function is also arbitrary and a fixed length of output (the output's length is from 128 bits).
- *Low computational cost* - the cost of computation for any inputs is cheap.

- *One-way hash function* - there are two requirements of this condition. Firstly, with a given output $h(x) = y$, it becomes too really difficult to find back the input x. In addition, given two values x and $h(x)$, it is also difficult to find another x' with $x' \neq x$ and $h(x') = h(x)$.
- *Collision resistant* - the difficulty is to find two distinct inputs which can obtain the same hash value or result.

2.2 Collision free hash function

Another case is about *the collision free* [4] which mentions a condition that the hash function can shift a message with any length to a fixed length string. Moreover, with different messages for input, we should have distinct two outputs. To be clear about this concept, there is a set of requirements for a fixed size collision-free hash function family F:

- The probabilistic polynomial with $\Theta(m)$ is to choose an instance $f \in F$ with m being the size of collision free hash function family.
- The computation with any messages for input should take a polynomial time for any instance in F.
- Given a hash function $f \in F$, it should definitely hard to find two distinct messages x, y such that $f(x) \neq f(y)$.

2.3 What makes a good hash function?

To be a good hash function, there is a requirement called simple uniform hashing. This requirement is about the independence of hash values from different keys and each key is hashed into a hash value belonging to a range with equal distribution. [5] mentioned that is too hard to check this condition because of the difficulty in probability distribution from hashed keys. In other words, a hash function will be considered as a good one if it can minimize the opportunity that slightly different keys are hashed into the same hash value. Moreover, a hash function is expected to obtain hash values which are independent of any patterns in the considered data.

3 Universal Hashing

The main idea of universal hashing [3] is to randomly choose the hash function which is also independent of the keys. Once using universal hashing, the hash functions become more flexible than fixed hash functions, which reduces the vulnerabilities of malicious adversaries. In particular, the adversaries can select n keys to obtains n hash values from a fixed hash function in order to aim the same slot or hash value with the average time being $\Theta(n)$.

In term of universal hashing, due to the hash function going to become random, the algorithm can be different from each operation in order to achieve good performance for inputs (also with the same key inputted). Meanwhile, the poor performance can happen if the random hash function poorly causes the set of identifiers to hash.

A finite collection of hash functions is considered as H, whereas let say U is the universe of keys into the range $\{0, 1, 2, .., m-1\}$ for the collection H. The collection is called universal if for each $k, l \in U$ and the number of hash functions $h \in H$ such that $h(k) = h(l)$ being the most $\frac{|H|}{m}$. It particularly can say that with an arbitrary hash function from H, the probability for a collision between two distinct keys h, l is not higher than $\frac{1}{m}$ of a collision in the case of $h(k)$ and $h(l)$ independently and randomly chosen from U.

Let consider a universal collection of hash functions to hash n keys into a table T with size m. In this case, a key k does not belong to the table, so the expectation $E[n_{h(k)}]$ of table's length is at most the load factor n/m. Otherwise, the expectation $E[n_{h(k)}]$ of the list is at most $1 + \alpha$. Note that this expectation is based on the choice of the hash function with any keys. Therefore, with a pair distinct k, l keys, an indicator random variable $X_{kl} = I[h(k) = h(l)]$ (that means $I[h(k) = h(l)] = 1$ if $h(k) = h(l)$; otherwise, $I[h(k) = h(l)] = 0$). Since this pair k, l can collide with the maximum probability at $1/m$, it can be said that $Pr[h(k) = h(l)] \leq 1/m$. It hence can say that $E[X_{h(k)}] \leq 1/m$. On the other hand, a random variable Y_k is the number of keys which are different from k. Thus, this random variable is $Y_k = \sum_{\substack{l \in T \\ l \neq k}} X_{kl}$. Let take the expected values from this equation, we can have $E[Y_k] = E[\sum_{\substack{l \in T \\ l \neq k}} X_{kl}]$. Then, with the linearity of expectation, the right-side of the equation can become $\sum_{\substack{l \in T \\ l \neq k}} E[X_{kl}]$ which is less or equal to $\sum_{\substack{l \in T \\ l \neq k}} \frac{1}{m}$. Finally, if we have $k \notin T$, $n_{h(k)} = Y_k$ and $|l : l \in T,\, l \neq k| = n$. As a consequence, $E[n_{h(k)}] = E[Y_k] \leq n/m$. In the case $k \in T$, it means that key k can be found in table $T[h(k)]$ and Y_k cannot contain key k, so $n_{h(k)} = Y_k + 1$ and $|l : l \in T,\, l \neq k| = n - 1$. As a result, we can have

$$E[n_{h(k)}] = E[Y_k] + 1 \leq \frac{n-1}{m} + 1 = 1 + \frac{n}{m} - \frac{1}{m} \leq \frac{n}{m} + 1$$

In another case, if we utilize universal hashing and collision resolution with chaining on a table of size m, it can obtain an expected time $\Theta(n)$ in order to execute a given sequence operations' size n (INSERT, SEARCH and DELETE) with $\Theta(m)$ INSERT operation. Because the number of INSERT operations is $O(m)$, if we have $n = O(m)$, we can say $n/m = O(1)$. Meanwhile, due to the SEARCH and DELETE operations executing in a fixed time, it can say SEARCH operation getting $O(1)$. As a consequence, with the linearity of expectation, the expected time for the entire sequence of n operations is $O(n)$, so $\Theta(m)$ will bound it.

Next, it is interesting in answering the question that "How can design a universal class of hash functions?". Following [5], in the beginning, a large prime number p which can be bound every key k. Z_p and Z_p^* represents the set $\{0, 1, ..., p-1\}$ and $\{1, ..., p-1\}$, respectively. Additionally, let assume the size of the universe of keys is higher than the size of the hash table, so we can say

$p > m$. Now, we call h_{ab} being a hash function with $a \in Z_p^*$ and $b \in Z_p$. This hash function can be $h_{ab}(k) = ((ak+b)\, mod p)\, mod m$. In this case, the family of all such hash function is $H_{pm} = \{h_{ab} : a \in Z_p^* \, and \, b \in Z_p\}$. [5] mentioned that this design can obtain a property which is the range of the output becoming arbitrary. Moreover, this design generates H_{pm} with $p(p-1)$ hash functions since a and b can have $p-1$ and p choices, in turn.

Let see with two distinct keys k, l $(k \neq l)$ in Z_p and a given hash function h_{ab}, we can obtain these values: $r = (ak+b)\; mod\, p$ and $s = (al+b)\; mod\, p$. Obviously, we can say that $r \neq s$ because of $r - s \equiv a(k-l)\, mod p$ which cannot be zero $(k \neq l)$, which leads to no collisions at the $mod\, p$. In addition, with $p(p-1)$ different choices from the pair (a, b), it can generate different pair (r, s) with $r \neq s$. To prove it, we can consider to solve $a\, and\, b$ with a given pair (r, s), so $a = [(r-s)((k-l)^{-1}\; mod\, p)]\; mod\, p$ and $b = (r - ak)\; mod\, p$. Thus, with two distinct inputs k, l and a pair (a, b) - uniformly at random, any pair (r, s) will be different after modulo p. It also can be said that there is a one-to-one mapping between k, r and l, s.

The probability of two distinct keys k, l colliding is to become the probability which $r \equiv s(mod\, m)$ with $r\; and\; s$ being random from the modulo p. Therefore, we can have $p-1$ values for s with a given r. At this point, we have $\lceil \frac{p}{m} \rceil - 1 \leq ((p+m-1)/m) - 1 = (p-1)/m$, so the maximum probability to get $r \equiv s$ in modulo m is $\frac{(p-1)/m}{p-1} = \frac{1}{m}$. From this point of view, we can say that with two distinct keys k, l in Z_p, the probability of collision can be less or equal to $1/m$ or $Pr[h_{ab}(k) = h_{ab}(l)] \leq 1/m$.

4 Perfect Hashing

A perfect hashing [6] is required two levels of hashing in which level a universal hashing is utilized. At the first level, we consider hashing n keys into m slots through a hash function h generated by a family of universal hash functions. In the other level, at the slot j, another hash function h_j is utilized to avoid collisions at this level.

From this view, we can see that there are two things need to be considered. Firstly, the use of the function h_j for bucket j which has n_j entries (from the result of hash function h) with $m = n_j^2$. Therefore, it requires a probability to prove that h_j should be perfect (this probability $\geq 1/2$). In another concern, the space to contain these keys is expected to be $O(n)$ instead of a huge amount of space or $\sum_{j=0}^{m-1} n_j^2 = O(n)$.

To explain the first issue, to prove h_j being perfect (its probability is greater than 1/2), we have a indicator random variable X_{kl} being 1 if $h(k) = h(l)$ with $h \neq l$ (the probability of this choice is $1/m$), while this variable is zero in other cases. It hence can have the number of collisions $X = \sum_{k \neq l} X_{kl}$. As a consequence, the probability that we do not have any collisions is $Pr[X = 0] =$

$1 - Pr[X \geq 1] > \frac{1}{2}$. In particular, we have an expectation being

$$\begin{aligned}E[X] = \sum_{k \neq l} E[X_{kl}] &= \binom{n_j}{2} E[X_{kl}] \\ &= \binom{n_j}{2} \frac{1}{m} = \binom{n_j}{2} \frac{1}{n_j^2} \\ &= \frac{n_j(n_j - 1)}{2} \frac{1}{n_j^2} \\ &= \frac{n_j - 1}{2n_j} < \frac{1}{2}\end{aligned}$$

At this point, the Markov's inequality can be applied to generate $Pr[X \geq 1] \leq \frac{1/2}{1} = \frac{1}{2}$.

In the second concern, we desire to prove that to store n keys in a hash table size $m = n$ with a hash function h from a universal class of hash functions, we have $E[\sum_{j=0}^{m-1} n_j^2] < 2n$ (n_j the number of keys needs to store in slot j). Let start this from the expectation $E[\sum_{j=0}^{m-1} n_j^2]$, we can generate the following equations based on $a^2 = a + 2\binom{a}{2}$.

$$E[\sum_{j=0}^{m-1} n_j + 2\binom{n_j}{2}] = E[\sum_{j=0}^{m-1} n_j] + 2E[\sum_{j=0}^{m-1} \binom{n_j}{2}]$$

The right hand side can separate into two parts. The first part $E[\sum_{j=0}^{m-1} n_j]$ can be $E[n]$, while the second part $2E[\sum_{j=0}^{m-1} \binom{n_j}{2}]$ is considered as the number of collisions in bucket j, which means

$$E[\sum_{j=0}^{m-1} \binom{n_j}{2}] = E[\sum_{k \neq l} X_{kl}] = \binom{n}{2} \frac{1}{m} = \binom{n}{2} \frac{1}{n} = \frac{n-1}{2}$$

Finally, we can summarize that

$$E[\sum_{j=0}^{m-1} n_j + 2\binom{n_j}{2}] = n + n - 1 < 2n$$

However, in the term of the probability $Pr[\sum_{j=0}^{m-1} n_j^2 \geq 4n]$, we can applied Markov's inequality again to get

$$Pr[\sum_{j=0}^{m-1} n_j^2 \geq 4n] \leq \frac{Pr[\sum_{j=0}^{m-1} n_j^2]}{4n} = \frac{2n}{4n} = \frac{1}{2}$$

5 Count-Min Sketch

The Count-Min or CM sketch [7] is named based on two operations (counting and calculating the minimum). In the beginning, the data structure of the CM sketch with two parameters (ϵ, δ) is a two-dimensional array. Let say the size of the array is w as width and d as depth. From the definition, with pair parameter (ϵ, δ), the array' size is $w = \lceil e/\epsilon \rceil$ and $d = \lceil ln(1/\delta) \rceil$. Moreover, initially, every entry's value in the array starts at zero. The number of hash functions is equal to the depth d of the array, so it can be said that $h_1 ... h_d : \{1...n\} \longrightarrow \{1...w\}$. In other words, with a hash function (based on the depth of the array), any keys obtain a hash value which indicates to a value in the range $\{1...w\}$ of the array's width.

The UPDATE procedure is next considered with a request (i_t, c_t), which means an item a_{i_t} needs to update as addition a new value c_t. Therefore, it can be said that $\forall j, 1 \leq j \leq d$, $count[j, h_j(i_t)] \longleftarrow count[j, h_j(i_t)] + c_t$. From this point, a set of universal hash functions are independently chosen from $H : h_1...h_d$. These hash functions are calculated as the UPDATE or COUNT procedure before taking the value $j \in \{1, ..., d\}$ to obtain the minimum hash value $\hat{a}$ from the universal hash functions' result ($\hat{a} = min_j count\lceil j, h_j(i) \rceil \leq a_i + X_{ji}$) with $X_{ji} \geq 0$ and finding j to minimize X_{ji}. An indicator variable $I_{i,j,k}$ is 1 if $(i \neq k) \wedge (h_j(i) = h_j(k))$ and this value is 0 in other cases. Therefore, we can have $X_{ji} = \sum_{k=1}^{n} I_{jik} a_k$ which implies

$$E[X_{ji}] = \sum_{k=1}^{n} E[I_{jik} a_k] = \sum_{k=1}^{n} a_k E[I_{jik}] = \frac{\epsilon}{e} \sum_{k=1}^{n} a_k = \frac{\epsilon}{e} ||a||_1$$

The last equation is from $E[I_{jik}] = Pr[I_{jik}] = 1) = 1/w = \epsilon/e$. Then, we can obtain the probability of $\hat{a} > a_i + \epsilon||a||_1$ as below:

$$\begin{aligned}
Pr[\hat{a} > a_i + \epsilon||a||_1] &= Pr[\forall j,\ count[j, h_j(i)] > a_i + \epsilon||a||_1] \\
&= Pr[\forall j,\ a_i + X_{ji} > a_i + \epsilon||a||_1] \\
&= Pr[\forall j,\ X_{ji} > \epsilon||a||_1] \\
&= \prod_{j=1}^{d} Pr[X_{ji} > \epsilon||a||_1] \\
&< \prod_{j=1}^{d} \frac{E[X_{ji}]}{2} (Markov's\ inequality) \\
&= \prod_{j=1}^{d} \frac{1}{e} < \prod_{j=1}^{d} \frac{1}{2} = \frac{1}{2^d} = \delta.
\end{aligned}$$

Finally, it can be summarized that $a_i \leq \hat{a}_i$, while with the probability at least $1 - \delta$, we can have $\hat{a}_i \leq a_i + \epsilon||a||_1$.

6 Conclusion

The assignment is to study hash functions which are based on the randomized algorithms. Particularly, the study on hash functions includes universal hashing with a random choice of a hash function in a set of universal class hash functions. Furthermore, the perfect hashing is described as two layers of universal hashing with analysis of its probability from collisions and space. Finally, vthe count-min sketch is analyzed through their probability of collisions.

7 Acknowledgement

I would like to express my sincere gratitude to Prof. Juha Kortelainen and Dr. Juha Partala through lectures in this special course "Randomized Algorithms" at the University of Oulu.

References

[1] B. Preneel, *Analysis and design of cryptographic hash functions.* PhD thesis, Citeseer, 1993.

[2] R. C. Merkle, "One way hash functions and des," in *Conference on the Theory and Application of Cryptology*, pp. 428–446, Springer, 1989.

[3] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pp. 33–43, ACM, 1989.

[4] I. B. Damgård, "A design principle for hash functions," in *Conference on the Theory and Application of Cryptology*, pp. 416–427, Springer, 1989.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms.* MIT press, 2009.

[6] M. L. Fredman, J. Komlós, and E. Szemerédi, "Storing a sparse table with o (1) worst case access time," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pp. 165–169, IEEE, 1982.

[7] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.