Lecture 4: Hashing

Lecturer: Rong Ge, Scribe: Weiyao Wang

January 29, 2018

1 Lecture Overview

Last time we talked about some techniques that are frequently used in analyzing randomized algorithms. In this lecture, we are going to see one application of randomized algorithm: Hashing

2 Background Knowledge

2.1 Definition

Hash table is a kind of data structure for set with the following properties:

- Elements in the set: subset of $\{0, 1, 2, ..., m-1\}$, where m is the largest possible elements in the set and m is very large. For example, $m = 2^{32}$ or 2^{64}.
- Elements in the set can also be other data structures. For example, we may map any data structure to integer, and then map integer to elements in the set.
- There are three operations involved for Hashing: Insert, Delete, Find.

In this lecture, we will primarily focus on integers as elements of the set.

2.2 Goal for Hashing

We have two goals for Hashing listed below: the first is run time goal and the second is space goal.

1. All operations (Insert, Delete, Find) are in $O(1)$ time (in expectations).
2. If the hash table has k elements where $k << m$, then the hash table takes $O(k)$ space.

2.3 Idea of Hashing

The idea of hashing is to put elements into random locations in the hash table.

For example, we are given a hash table with size n:

0	1	...	$n-1$

We put element $i \in [m]$ (where $[m]$ stands for $\{1, ..., m\}$) into a "random" location in $[n]$.

Hash Function h is thus defined to be a mapping: $[m] \to [n]$. Element $i \in [m]$ will be mapped to $h(i) \in [n]$.

For example: $h(35) = 3$, $h(427) = 5$, $h(1025) = n - 1$, we have the hash table becomes:

0	1	2	3	4	5	...	$n-1$
			35		427	...	1025

2.4 Challenge

The challenge for Hashing is that since $m >> n$, it's impossible to insert every element $\in [m]$ into a unique place in the hash table (i.e. h cannot be injective). Therefore, we need to handle collision (two elements mapped to the same place).

To handle collision, we let each location maintains a linked list. We hope that the linked list is short to keep the run time for Delete/ Find to be $O(1)$. Therefore, we need to choose the hash function h wisely.

3 Two Problematic Approaches

3.1 Choosing a fixed function h

For any $h : [m] \to [n]$, we can find $x_1, ..., x_k$ such that $h(x_1) = h(x_2) = ... = h(x_k)$ (as long as $kn < m$).

If the input is $x_1, ..., x_k$, the linked list at $h(x_i)$ has length k. Therefore, the hash table essentially degenerates to a linked list, and thus operations (such as Find) may take $O(k)$ time.

Still, this fixed function approach has applications in areas like cryptography, where the input domain is too large and collisions will be rare.

Note that analyzing this approach using average case analysis may yield to better results for the algorithm, but when analyzing randomized algorithm, we are considering worst-case scenario.

3.2 Choosing a function uniformly at random

The total number of functions mapping from $[m] \to [n]$ is given by n^m, where each element in $[m]$ may be mapped to any element of $[n]$. Therefore, the number of bits required to store such function is given by $log(n^m) = mlog(n) >> O(k)$, the space requirement.

4 Pairwise Independence Hashing

4.1 Pairwise Independence

Given a sequence of random variables $X_1, ..., X_n$, we say they are pairwise independent if $\forall i \neq j$, X_i is independent of X_j.

A remark here is that Pairwise Independence is a less strong condition compared to independence.

For example, let X_1 and X_2 be two independent variables with $Pr(X_1 = 1) = Pr(X_2 = 1) = Pr(X_1 = 0) = Pr(X_2 = 0) = o.5$. Let $X_3 = X_1 \oplus X_2 = (X_1 + X_2) mod 2$. Then the possible values for the tuple (X_1, X_2, X_3) are $\{(0,0,0), (0,1,1), (1,0,1), (1,1,0)\}$, each with probability $\frac{1}{4}$.

In this example, it is easy to verify that X_1, X_2, X_3 are pairwise independent. And we can also verify that they are not independent of each other.

4.2 Pairwise Independent Hash Family

A hash family is defined as a set of hash functions, denote by $\mathbb{H}$.

For any $x \neq y \in [m]$ and $u, v \in [n]$, we hope that

$$Pr_{h \in \mathbb{H}}[h(x) = u, h(y) = v] = \frac{1}{n^2} (\approx \frac{1}{n^2})$$

4.3 Constructing Pairwise Independent Hash Family

Recall: Modulo Arithmetic

Given discrete field $\mathbb{Z}_p : \{0, 1, 2, ..., p-1\}$, where p is prime, we preserve the four basic operations: $+, -, \times, \div$. From abstract algebra, we know that for any $x \neq 0 (mod\ p)$, there is a unique element $y \in \mathbb{Z}_p$ such that $xy = 1 (mod\ p)$, we write $y = x^{-1}$.

We choose $p > m >> n$ such that p is prime. We write $H = \{(a, b) | a, b \in \mathbb{Z}_p\}$. We define function $f : \mathbb{Z}_p \to \mathbb{Z}_p$ by $f_{a,b}(x) = (ax + b) (mod\ p)$. We further define $h_{a,b}(x) = f_{a,b}(x) (mod\ n)$.

With the constructions above, we prove the following properties:

Lemma: $\{f_{a,b} | (a, b) \in \mathbb{Z}_p\}$ is a pairwise independent hash family from $\mathbb{Z}_p \to \mathbb{Z}_p$.

Proof: We want to prove that $\forall x \neq y \in \mathbb{Z}_p$ and $\forall u, v \in \mathbb{Z}_p$, we have

$$Pr_{a,b \in \mathbb{Z}_p}[f_{a,b}(x) = u, f_{a,b}(y) = v] = \frac{1}{p^2}$$

We may re-write the probability (left-hand side) as

$$Pr = \frac{\text{number of pairs } (a, b) \text{ that satisfy } f_{a,b}(x) = u, f_{a,b}(y) = v}{p^2}$$

This gives us the following requirement:

- $ax + b = u(mod\ p)$
- $ay + b = v(mod\ p)$

Since the arithmetic operations in the integer fields have correspondence in $\mathbb{Z}_p$, we may treat the above constraints as a system of linear equations, with solutions:

- $a = (u - v)(x - y)^{-1}$
- $b = u - ax = v - ay$

Note that since all operations given unique outputs, the solution above is unique. Therefore the number of pairs is 1. QED

The Lemma can be extended to proving $\{h_{a,b}|(a, b) \in \mathbb{Z}_p\}$ is also a pairwise independent hash family:

$\forall x \neq y \in [m]$ and $\forall u, v \in [n]$, we have

$$\begin{aligned} Pr_{a,b\in\mathbb{Z}_p}[h_{a,b}(x) = u, h_{a,b}(y) = v] &= \sum_{i,j} Pr_{a,b\in\mathbb{Z}_p}[f_{a,b}(x) = u + in, f_{a,b}(y) = v + jn] \\ &= \sum_{i,j} \frac{1}{p^2} \\ &\approx (\frac{p}{n})(\frac{p}{n})\frac{1}{p^2} = \frac{1}{n^2} \qquad (1) \end{aligned}$$

where (1) is true since the number of i or j to satisfy the requirements is $\approx \frac{p}{n}$.

4.4 Analysis: Expected Run Time for Find Operation

Setting: given $x_1, ..., x_k$ are already in the hash table, we would like to know the run time for Find(y).

The run time would be approximately the length of linked list at $h(y)$.

If $x_i = y$, then $h(x_i) = h(y)$. Else, $Pr[h(x_i) = h(y)] = \frac{1}{n}$. Therefore, we have

$$\begin{aligned} \mathbb{E}[\text{length of linked list}] &= \mathbb{E}[\text{number of } x_i \text{ s.t. } h(x_i) = h(y)] \\ &\leq 1 + \frac{1}{n}(k - 1) \leq 1 + \frac{k}{n} \end{aligned}$$

which is a constant

4.5 Analysis: $k = \sqrt{n}$, what is the probability of having at least one collision?

Let $A_{i,j} = 1$ if $h(x_i) = h(x_j)$, the event that a particular collision happens. Using the same logic we had analyzing Balls and Bins problem, we have $\mathbb{E}[A_{i,j}] = \frac{1}{n}$. Then $\mathbb{E}[\text{number of collisions}] = \sum_{1 \le i < j \le n} \mathbb{E}[A_{i,j}] = \frac{k(k-1)}{2n} \le \frac{1}{2}$

4.6 Remark on k-wise independence

k-wise independence of variables is defined by the independence of any subset of variables with size at most k.

We can construct other types of hash functions. For example, for 3-wise independence, we may define $h_{a,b,c} = ax^2 + bx + c(mod\ p)$. If $k \to logn$, k-wise would not be too much different from actual independence of all.