Advanced Algorithm Design: Hashing

Lectured by Prof. Moses Charikar
Transcribed by Linpeng Tang*

Feb 2nd, 2013

1 Preliminaries

In hashing, we want to store a subset S of a large universe U (U can be very large, say $|U| = 2^{32}$ is the set of all 32 bit integers). And $|S| = m$ is a relatively small subset. For each $x \in U$, we want to support 3 operations:

- *insert*(x). Insert x into S.
- *delete*(x). Delete x from S.
- *query*(x). Check whether $x \in S$.

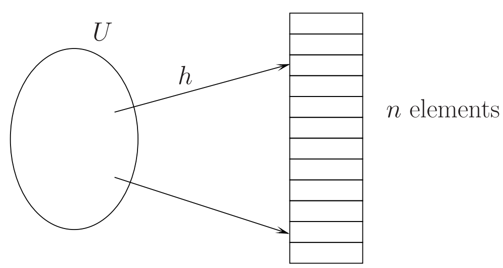

Figure 1: Hash table. x is placed in $T[h(x)]$.

A hash table can support all these 3 operations. We design a hash function

$$h : U \longrightarrow \{0, 1, \ldots, n-1\} \tag{1.1}$$

such that $x \in U$ is placed in $T[h(x)]$, where T is a table of size n.

Since $|U| \gg n$, multiple elements can be mapped into the same location in T, and we deal with these collisions by constructing a linked list at each location in the table.

One natural question to ask is: how long is the linked list at each location?

We make two kinds of assumptions:

*linpengt@cs.princeton.edu

1. Assume the input is the random.
2. Assume the input is arbitrary, but the hash function is random.

Assumption 1 may not be valid for many applications, since the input might be correlated.

For Assumption 2, we construct a set of hash functions $\mathcal{H}$, and for each input, we choose a random function $h \in \mathcal{H}$ and hope that on average we will achieve good performance.

2 Hash Functions

Say we have a family of hash functions $\mathcal{H}$, and for each $h \in \mathcal{H}$, $h : U \longrightarrow [n]$[1], what do mean by saying these functions are random?

For any $x_1, x_2, \ldots, x_m \in S$ ($x_i \neq x_j$ when $i \neq j$), and any $a_1, a_2, \ldots, a_m \in [n]$, ideally a random $\mathcal{H}$ should satisfy:

- $\Pr_{h\in\mathcal{H}}[h(x_1) = a_1] = \frac{1}{n}$.
- $\Pr_{h\in\mathcal{H}}[h(x_1) = a_1 \wedge h(x_2) = a_2] = \frac{1}{n^2}$. Pairwise independence.
- $\Pr_{h\in\mathcal{H}}[h(x_1) = a_1 \wedge h(x_2) = a_2 \wedge \cdots \wedge h(x_k) = a_k] = \frac{1}{n^k}$. k-wise independence.
- $\Pr_{h\in\mathcal{H}}[h(x_1) = a_1 \wedge h(x_2) = a_2 \wedge \cdots \wedge h(x_m) = a_m] = \frac{1}{n^m}$. Full independence (note that $|U| = m$). In this case we have n^m possible h (we store $h(x)$ for each $x \in U$), so we need $m \log n$ bits to represent the each hash function. Since m is usually very large, this is not practical.

For any x, let L_x be the length of the linked list containing x, then L_x is just the number of elements with the same hash value as x. Let random variable

$$I_y = \begin{cases} 1 & \text{if } h(y) = h(x), \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

So $L_x = 1 + \sum_{y\neq x} I_y$, and

$$E[L_x] = 1 + \sum_{y\neq x} E[I_y] = 1 + \frac{m-1}{n} \tag{2.2}$$

Note that we don't need full independence to prove this property, and pairwise independence would actually suffice.

[1] We use $[n]$ to denote the set $\{0, 1, \ldots, n-1\}$

3 2-Universal Hash Families

Definition 3.1 (Cater Wegman)**.** Family $\mathcal{H}$ of hash functions is 2-universal if for any $x \neq y \in U$,

$$\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{n} \tag{3.1}$$

Note that this property is even weaker than 2 independence.

We can design 2-universal hash families in the following way. Choose a prime $p \in \{|U|, \ldots, 2|U|\}$, and let

$$f_{a,b}(x) = ax + b \mod p \qquad (a, b \in [p], a \neq 0) \tag{3.2}$$

And let

$$h_{a,b}(x) = f_{a,b}(x) \mod n \tag{3.3}$$

Lemma 3.2. *For any $x_1 \neq x_2$ and $s \neq t$, the following system*

$$ax_1 + b = s \mod p \tag{3.4}$$

$$ax_2 + b = t \mod p \tag{3.5}$$

has exactly one solution.

Since $[p]$ constitutes a finite field, we have that $a = (x_1 - x_2)^{-1}(s - t)$ and $b = s - ax_1$. Since we have $p(p-1)$ different hash functions in $\mathcal{H}$ in this case,

$$\Pr_{h \in \mathcal{H}}[h(x_1) = s \wedge h(x_2) = t] = \frac{1}{p(p-1)} \tag{3.6}$$

Claim 3.3. *$\mathcal{H} = \{h_{a,b} : a, b \in [p] \wedge a \neq 0\}$ is 2-universal.*

Proof. For any $x_1 \neq x_2$,

$$\Pr[h_{a,b}(x_1) = h_{a,b}(x_2)] \tag{3.7}$$

$$= \sum_{s,t \in [p], s \neq t} \delta_{(s=t \mod n)} \Pr[f_{a,b}(x_1) = s \wedge f_{a,b}(x_2) = t] \tag{3.8}$$

$$= \frac{1}{p(p-1)} \sum_{s,t \in [p], s \neq t} \delta_{(s=t \mod n)} \tag{3.9}$$

$$\leq \frac{1}{p(p-1)} \frac{p(p-1)}{n} \tag{3.10}$$

$$= \frac{1}{n} \tag{3.11}$$

where δ is the Dirac delta function. Equation (3.10) follows because for each $s \in [p]$, we have at most $(p-1)/n$ different t such that $s \neq t$ and $s = t \mod n$. $\square$

Can we design a collision free hash table then? Say we have m elements, and the hash table is of size n. Since for any $x_1 \neq x_2$, $\Pr_h[h(x_1) = h(x_2)] \leq \frac{1}{n}$, the expected number of total collisions is just

$$E[\sum_{x_1 \neq x_2} h(x_1) = h(x_2)] = \sum_{x_1 \neq x_2} E[h(x_1) = h(x_2)] \leq \binom{m}{2} \frac{1}{n} \tag{3.12}$$

Let's pick $m \geq n^2$, then

$$E[\text{number of collisions}] \leq \frac{1}{2} \tag{3.13}$$

and so

$$\Pr_{h \in H}[\exists \text{ a collision}] \leq \frac{1}{2} \tag{3.14}$$

So if the size the hash table is large enough $m \geq n^2$, we can easily find a collision free hash functions. But in reality, such a large table is often unrealistic. We may use a two-layer hash table to avoid this problem.

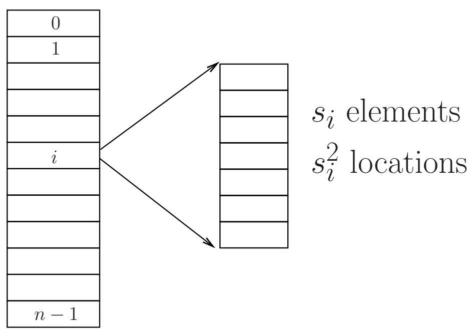

Figure 2: Two layer hash tables.

Specifically, let s_i denote the number of collisions at location i. If we can construct a second layer table of size s_i^2, we can easily find a collision-free hash table to store all the s_i elements. Thus the total size of the second-layer hash tables is $\sum_{i=0}^{m-1} s_i^2$.

Note that $\sum_{i=0}^{m-1} s_i(s_i - 1)$ is just the number of collisions calculated in Equation (3.12), so

$$E[\sum_i s_i^2] = E[\sum_i s_i(s_i - 1)] + E[\sum_i s_i] = \frac{m(m-1)}{n} + m \leq 2m \tag{3.15}$$

4 Load Balance

In load balance problem, we can imagine that we are trying to put balls into bins. If we have n balls and n bins, and we randomly put the balls into bins, then for a give i,

$$\Pr[\text{bin}_i \text{ gets more than } k \text{ elements}] \le \binom{n}{k} \cdot \frac{1}{n^k} \le \frac{1}{k!} \tag{4.1}$$

By Stirling's formula,

$$k! \sim \sqrt{2nk}(\frac{k}{e})^k \tag{4.2}$$

If we choose $k = O(\frac{\log n}{\log \log n})$, we can let $\frac{1}{k!} \le \frac{1}{n^2}$. Then

$$\Pr[\exists \text{ a bin} \ge k \text{ balls}] \le n \cdot \frac{1}{n^2} = \frac{1}{n} \tag{4.3}$$

So with probability larger than $1 - \frac{1}{n}$[2],

$$\max \text{load} \le O(\frac{\log n}{\log \log n}) \tag{4.4}$$

Note that if we look at 2 random bins when a new ball comes in and put the ball in the bin with fewer balls, we can achieve maximal load at the scale of $O(\log \log n)$, which is a huge improvement.

[2]this can be easily improve to $1 - \frac{1}{n^c}$ for any constant c