Hashing

Huy L. Nguyễn

In this note, we will study hashing, a fundamental data structure in computer science. This is also a good setting to develop some proficiency with probability, which will become useful later.

Hashing is a solution to the *dictionary* problem. In fact, the solution has become so synonymous with the problem that in some programming languages, a hash table is called a "dictionary". In the dictionary problem, we would like to maintain a dynamic database of up to n keys where we can insert and delete keys and quickly look up the associated data of any given key.

Formally, the data structure needs to store a subset S of a large universe U. The size of S, denoted by $|S| = n$, is typically much smaller than the size of U. For each key $x \in U$, we would like to support 3 operations:

- *insert*(x): insert x into S.
- *delete*(x): delete x from S.
- *lookup*(x): check if x is in S.

The approach of hashing is to map objects from the universe U to a hash table with m slots, with m much smaller than the size of U:

$$h : U \to [m]$$

where $[m]$ denotes the set $\{0, 1, 2, \ldots, m-1\}$. In general, there can be two keys $x, y \in U$ that are mapped to the same slot i.e. $h(x) = h(y)$. This is called a *collision*.

One way to deal with collision is *hashing with chaining*: for each slot in the hash table, we store a linked list of keys that are mapped to that slot. When we look up a key x, we can go to the slot $h(x)$ and search for x in the linked list at that slot.

Question: what is the running time of *lookup*(x)?

The running time of *lookup*(x) is proportional to the length of the linked list at $h(x)$. Thus, for the algorithm to be fast, it is important to make the linked lists short.

Question: the keys we would like to hash are 128-bit integers. Should we just use MD5 (a public deterministic cryptographic hash function)?

While there is no single hash function that is good for all inputs, the approach we take will be to designate a set of hash functions $\mathcal{H}$ and when we need to hash a fixed input S, we will pick a random function $h \in \mathcal{H}$ and hope that on average, we will achieve a good performance.

The performance of a hash function is governed by the following three key metrics:

- *Speed* the time it takes to compute $h(x)$
- *Space* the size of the random seed to select h from the family $\mathcal{H}$
- *Relation of the hash values $h(x)$ for different x's*

1 Universal hashing

As we saw above, the speed of *lookup*(x) depends on the number of collisions x has in the table. Thus, a desirable property of the hash table is that for any two keys x, y, the probability that they collide is low. This consideration motivates the following definition.

Definition 1.1 (Carter and Wegman '79). *A family $\mathcal{H}$ is* universal *if for any two distinct keys x and y in U,*

$$\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \le \frac{1}{m}$$

For many applications, it suffices to have a slightly weaker guarantee that

$$\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \le \frac{c}{m}$$

for some small constant c. This property is called *c-universal.*

Question: For $m > |U|$, is the family with just the identity function $h(x) = x$ a universal hash family?

Question: If a hash family $\mathcal{H}$ has collision probability 0, how large must m be?

2 Probabilty review

As we saw, the running time of look up depends on the number of keys with same hash value. Let's see what we can show assuming that the hash family is c-universal. This is also an opportunity to review some probability.

Let X be a random variable that takes value v with probability p_v. The *expectation* of X is defined as

$$\mathbb{E}[X] = \sum_v v \cdot p_v$$

Example: X is the number of heads in a fair coin toss. X takes value 1 with probability 1/2 and 0 with probability 1/2. The expectation of X is $\mathbb{E}[X] = 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2} = \frac{1}{2}$.

The linearity of expectation property states that

$$\mathbb{E}[X_1 + X_2] = \mathbb{E}[X_1] + \mathbb{E}[X_2]$$

This property is useful for analyzing the expectation of complicated random variable.

Example: X is the number of heads in 100 fair coin tosses. Let X_i be the number of heads in the ith coin toss. We have

$$\mathbb{E}[X] = \mathbb{E}[X_1] + \cdots + \mathbb{E}[X_{100}] = \frac{1}{2} \cdot 100 = 50$$

Let's analyze the expected number of keys that are hashed to the same value as $h(x)$. Let X_i be the random variable that takes value 1 if $h(i) = h(x)$ and value 0 otherwise. This is called an *indicator random variable.*

By c-universality, the probability that X_i takes value 1 is c/m. Therefore, by linearity of expectation, the expected number of keys with hash value $h(x)$ is

$$\frac{c}{m} \cdot n = \frac{cn}{m}$$

Question: what is the expected number of collision for a truly random function?

2.1 Markov's inequality

If X is a nonnegative random variable and $a > 0$ then Markov's inequality states that

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$$

Question: use Markov's inequality to bound the probability that we have at least $c\sqrt{n}$ keys with hash value 1.

3 A universal hash family

In this section, we will give an example of a universal hash family. Consider a prime $p \in [|U|, 2|U|]$. For each $a \in \{1, 2, \ldots, p-1\}$ and $b \in [p]$, define

$$f_{a,b}(x) = (ax + b) \bmod p$$
$$h_{a,b}(x) = f_{a,b}(x) \bmod m$$

Let $\mathcal{H}$ be the family of all the functions $h_{a,b}$. We will show that this family is universal. We will need the following lemma.

Lemma 3.1. *For any $x \neq y \in U$ and $s, t \in [p]$ the following system of equations with variables $a, b \in [p]$ has exactly one solution.*

$$(ax + b) \bmod p = s$$
$$(ay + b) \bmod p = t$$

Proof. The solution is $a = (y - x)^{-1}(t - s)$ and $b = s - ax$. □

We can now verify the definition of universality. Let $s = f_{a,b}(x)$ and $t = f_{a,b}(y)$. There are two possibilities for collision. The first case is when $s = t$. However, this can only happen if $a = 0$, which is not in our family.

Thus, collision can only happen when $s \neq t$ and $(s - t) \bmod m = 0$. For a fixed s, the number of choices for t leading to a collision is at most $\lceil p/m \rceil - 1 \leq (p-1)/m$. There are p choices for s so the number of pairs (s, t) leading to collision is at most $\frac{p(p-1)}{m}$.

For each pair (s, t), there is exactly one hash function with $f_{a,b}(x) = s, f_{a,b}(y) = t$. Thus, the number of pairs (a, b) leading to a collision is at most $\frac{p(p-1)}{m}$. There are $p(p-1)$ choices for (a, b) in our family.

Therefore, the probability of collision is at most

$$\frac{1}{p(p-1)} \cdot \frac{p(p-1)}{m} = \frac{1}{m}$$

Observe that this is exactly the collision probability for a uniformly random hash function. Thus, our universal hash family actually outperforms a truly random hash function!