

Universal Hashing and Perfect Hashing

Arpita Korwar

April 10, 2010

When Dumey introduced hashing as a solution to the dictionary problem, it had a heuristic flavour. The hash function would be chosen based on some assumptions. Then, in 1979, Carter and Wegman gave it a mathematical treatment. They presented the concept of universal hash functions.

The dictionary problem

We are given a series of operations: OP_1, OP_2, \dots . These operations are to be performed on an initially empty set S . Each operation OP_i can be one of the following:

- *Insert*(x) An item with key value x is inserted into the set S .
- *Lookup*(x) Check if an item with key value x is present in the set S .
- *Delete*(x) The item with key value x , if present, is deleted from the set S .

Each of the key values x comes from a universe U , i.e. $x \in U$. In this document, we assume $U = \{1, 2, \dots, N\}$.

Observe that the set S is a dynamic set. Each of the *Insert* and *Delete* operations may modify the set. Hence the size of the set S changes with each operation. We bound the maximum size of the set to n ($n \ll N$).

What are the data structures that can be used to store the set S ?

One option is to use a balanced binary search tree. But each of the operations would take $O(\log n)$ time. Moreover, a balanced binary search tree is more difficult to implement than, say, an array, or a singly linked list.

Could we store the set S in an array? Is there a data structure that would perform the above operations in constant time? Yes, there is such a data structure, hash table, which provides an easy way of storing such information.

Hash tables and hash functions

Let us denote the hash table by V . A *hash function* maps the elements of the universe to the hash table, $h : U \rightarrow V$. Let the size of the hash table be m . We would like the size of the hash table, m to be linear in the size of the set S . Obviously, the hash table cannot be as large as the universe of keys, N , in which case, there would be no collisions. When the hash table is smaller than the universe, collisions will occur. This is formalized by the following theorem.

Theorem 1. *If a hash function h is fixed and $N \geq m(n - 1) + 1$, then there exists set S , $|S| = n$, such that all the elements of S map to the same bucket of the hash table.*

Proof. By pigeon-hole principle. □

Thus, given any hash function, a malicious adversary can find a set S such that the number of collisions is maximum. To overcome this, we select a hash function randomly from a family of hash functions. This would ensure that any set selected by the adversary gets well distributed in expectation. This kind of randomization helps us overcome the adversary in the case of randomized quicksort also, where the pivot position is selected randomly.

One family of hash functions that distributes the set S well is the family of totally random functions, i.e. each element in U is mapped to any one bucket of the hash table with equal probability. The probability that key x is in bucket i is $1/m$. There are m^N such mappings.

Theorem 2. *When hash function h is chosen randomly from the set of all possible functions from U to V , the expected number of collisions is less than $\frac{n^2}{2m}$.*

Proof. Let

$$X_{ij} = \begin{cases} 1 & \text{if } h(i) = h(j) \text{ and } i \neq j; \\ 0 & \text{otherwise.} \end{cases}$$

Then $E[X_{ij}] = \Pr[h(i) = h(j)]$ when $i \neq j$. Since the hash function h is random, the probability that two keys in the set S collide is $\Pr[h(i) = h(j)] = \frac{1}{m}$, $i \neq j$. Let X be the total no. of collisions when set S is hashed to the

table using hash function h . Then by linearity of expectation,

$$\begin{aligned}
 X &= \sum_{i \neq j} X_{ij} \\
 \mathbb{E}[X] &= \mathbb{E} \left[\sum_{i \neq j} X_{ij} \right] \\
 &= \sum_{i \neq j} \mathbb{E}[X_{ij}] \\
 &= \sum_{i \neq j} \Pr[h(i) = h(j)] \\
 &= \sum_{i \neq j} \frac{1}{m} \\
 &= \binom{n}{2} \cdot \frac{1}{m} \\
 &\leq \frac{n^2}{2m}.
 \end{aligned}$$

□

But a random hash function needs $N \log m$ random bits. Computing and storing these many random bits is a costly affair. Hence, we would like to work with a smaller family of hash functions, which would need lesser number of random bits.

In 1979, Carter and Wegman made the following insightful observation. The only property of totally random functions that is used in the above proof is $\Pr[h(i) = h(j)] = \frac{1}{m}, i \neq j$. Thus, any family of hash functions for which the probability of collision is $\leq \frac{1}{m}$ will give the same number of collisions as a totally random family of hash functions, under expectation.

Definition 3 (Universal family of hash functions). *A family of hash functions from universe U to range $\{1, \dots, m\}$, is universal if $\forall x, y \in U$ and $x \neq y$, and $h \in_R \mathcal{H}$, $\Pr[h(i) = h(j)] = \frac{1}{m}$.*

Hence, if \mathcal{H} is a family of universal hash functions and $h \in_R \mathcal{H}$ then the expected number of collisions is $\mathbb{E}[X] \leq \frac{n^2}{2m}$.

Thus, using Markov's inequality, $\Pr \left[X \geq \frac{n^2}{m} \right] \leq \frac{1}{2}$. We will use this fact later when we study perfect hash functions.

Let us now look at a simple universal family of hash functions.

Example of universal hash function

Let the universe U be $\{1, \dots, N\}$ and let the hash table have buckets $\{1, \dots, m\}$. Let $b = \log m$ be the number of bits required to represent the key. Let \mathcal{H} be the universal family of hash functions. Each hash function is represented by a $b \times \log N$ matrix A of random bits. Then the hash function is given by $h(x) = Ax$. x is the key to be hashed, $x \in \{0, 1\}^b$.

e.g.: $Ax = h(x)$

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Theorem 4. If $x \neq y$, then $\Pr[h(x) = h(y)] = \frac{1}{m}$.

Proof. We can interpret the action of A on the key x as follows. Depending on the bit values of x , the columns of A are chosen to be XORed. i.e. if the i th bit of x , x_i is 1, only then, the i th column of A contributes to the value of $h(x)$.

$$\begin{aligned} h(x)_j &= \sum_{k=1}^{\log N} A_{jk}x_k \\ &= A_{ji}x_i + \sum_{k \neq i} A_{jk}x_k \end{aligned}$$

Now, since $x \neq y$, $\exists i$ such that $x_i \neq y_i$. Assume w.l.g. $x_i = 0, y_i = 1$. Suppose that the bits of all the columns of A , except for the i th column are computed.

Let us check under what conditions is $h(x) = h(y)$. This happens if and only if $\forall j, 1 \leq j \leq m, h(x)_j = h(y)_j$.

Since $x_i = 0$, the i th column of A does not contribute to $h(x)$. Hence, $h(x)$ has a fixed value, though the i th column of A is not fixed. The j th bit of $h(y)$ is given by

$$h(y)_j = A_{ji} + \sum_{k \neq i} A_{jk}y_k.$$

$\sum_{k \neq i} A_{jk}y_k$ has a fixed value once the corresponding columns of A are fixed. So, the value of $h(y)_j$ depends on the value of A_{ji} . $A_{ji} = 0$ or 1 with equal probability. So, $\Pr[h(y)_j = h(x)_j] = \frac{1}{2}$.

Hence, the probability that all the bits of $h(x)$ and $h(y)$ are same is $\Pr[h(y) = h(x)] = \frac{1}{2^b} = \frac{1}{m}$. \square

This universal hash function takes $\log N \log m$ random bits instead of $N \log m$ random bits. There are hash functions that use $O(\log N)$ random bits¹.

We now look at an application of universal hashing.

Perfect hashing

Once we find a data structure for storing dynamic data - hash table using universal hashing, a natural question is can we find an efficient data structure for storing a static set. We will now build a linear space data structure that takes constant lookup time (only for a static set).

We are given a universal family of hash functions. we pick a hash function from this family randomly. Recall from the section on hash tables and hash functions that the expected number of collisions, X , where n is the size of the set S and m is the size of the hash table is $E[X] \leq \frac{n^2}{2m}$ and $\Pr[X \geq \frac{n^2}{m}] \leq \frac{1}{2}$.

Lemma 5. *If $m = n^2$ then $\Pr[X \geq 1] \leq \frac{1}{2}$.*

So, when the size of the hash table is quadratic in the size of the set, the probability that there are no collisions is $> \frac{1}{2}$. We know that in Bernoulli trials, if success probability is p , then the expected number of trials before success is $\frac{1}{p}$. This immediately suggests a Las Vegas algorithm for finding a collision-free hash function. Pick a random hash function and check whether it is collision-free. If not, try again. Hence, we will find a collision-free hash function in expected 2 trials.

But the size of the hash table is quadratic. We want it to be linear.

Lemma 6. *If $m = n$ then $\Pr[X \geq m] \leq \frac{1}{2}$.*

Thus, in a linear sized hash table, there are $\leq m$ collisions with probability $> \frac{1}{2}$. With expected number of trials = 2, we can find such a hash function.

Our hash table would have two levels. At the first level, the hash table T has n buckets ($m = n$). We find the has function for this level such that there are $\leq m$ collisions. Now, at each bucket where collisions occur, we build second level hash tables with quadratic number of buckets. i.e. if

¹Check the book by Upfal and Mitzenmacher, chapter 13. Also refer the FKS algorithm.

$c_i > 1$ keys hash to bucket i , then we build a hash table T_i of size c_i^2 for that table. We find these hash functions such that there are no collisions.

To look up a key, we hash it to the table T . If there was a collision at the bucket to which it hashed, we use the hash function at the second level for that bucket. Hash tables at this level do not admit any collision. Thus, the key is found, if present in $O(1)$ time.

The total size of this two level table is $s = n + \sum_{i=1}^m c_i^2$. It now remains to show that this value is linear in n .

$$\begin{aligned} S &= n + \sum_{i=1}^m c_i^2 \\ &= n + \sum_{i=1}^m \left(2 \binom{c_i}{2} + c_i \right) \\ &= n + 2 \sum_{i=1}^m \binom{c_i}{2} + \sum_{i=1}^m c_i \end{aligned}$$

But $\binom{c_i}{2}$ = number of collisions at the i th bin. Since the total number of collisions $\leq m$, $\sum_{i=1}^m \binom{c_i}{2} \leq m = n$. Also, $\sum_{i=1}^m c_i$ = number of keys = n . So,

$$\begin{aligned} S &= n + 2n + n \\ &= 4n \end{aligned}$$

Hence, the two level hash table has $O(n)$ size.

References

- <http://www.brics.dk/Activities/98/pearls/bromille-un.ps>
- <http://www.cs.cmu.edu/~avrim/451f09/lectures/lect0929.pdf>
- Section 13.3: Families of Universal Hash functions from: "Probability and Computing" by Michael Mitzenmacher and Eli Upfal