COS 521: Advanced Algorithms Design February 5, 2004
Prof. Moses Charikar and Dr. Chandra Chekuri

Lecture 2: Universal and Perfect Hashing

Lecture: Prof. Moses Charikar *Scribe: Wolfgang Mulzer*

1 Why Universal Hashing?

In the last lecture, we defined the notion of a *2-universal hash family.* Now we are going to prove that the universal hashing scheme described in the last lecture actually yields constant expected running time for any sequence of operations on the hash table.

Theorem 1 *Let $R_1, R_2, \ldots, R_r$ be a sequence of operations on a hash table and assume that at most s of these operations are inserts. Furthermore, let H be a 2-universal family of hash functions. Then we get*

$$E_{h\in H}[t] \leq r(1 + \frac{s}{n}),$$

where t is the time needed to carry out the sequence of operations and n is the size of the hash table.

Proof: Each operation on the hash table first evaluates the hash function h and then operates on the linked list that is stored in the corresponding entry of the table (The only exception is MakeSet whose running time is constant). The time to compute the hash function is constant. Therefore, the running time of a single operation can be bounded by a constant plus the length of the linked list that is being manipulated. It follows that it suffices to bound the expected length of a linked list in the table, i. e. the number of elements that are mapped to the same location by h.

Fix an element $x \in S$ and let I_y be the indicator random variable for the event that $h(x) = h(y)$ for any $y \in S - \{x\}$. Since H is 2-universal, we have that $\Pr_{h\in H}[h(x) = h(y)] \leq \frac{1}{n}$ for any given $y \neq x$. Thus, the expected value of I_y is at most $\frac{1}{n}$. Consequently, the expected number of elements that collide with x is at most $\frac{s}{n}$, and the expected length of the list that contains x is at most $1+\frac{s}{n}$. As x was arbitrary, this upper bound holds for every linked list in the table, and hence the running time of a single operation can be bounded by $1 + \frac{s}{n}$. ■

Since we assume that $n = O(s)$, we get expected constant running time as promised.

2 A 2-universal Family of Hash Functions

Now that we have seen the power of universal hashing, the next question to ask is whether it is feasible. Or, to be more precise, is there a 2-universal family of hash-functions that can be described easily and that contains functions that can be computed efficiently? Fortunately, the answer to this question is yes.

Let p be a prime number such that $p \geq m$ where m is the size of the universe $\mathcal{U}$. Let $\mathbb{Z}_p := \{0, 1, \ldots, p-1\}$. For every $a, b \in \mathbb{Z}_p$ with $a \neq 0$, define functions $f_{ab} : \mathbb{Z}_p \to \mathbb{Z}_p$ by

$$f_{ab}(x) = ax + b \qquad (\text{mod } p).$$

Furthermore, for every $a, b \in \mathbb{Z}_p, a \neq 0$ define $h_{ab} : \mathbb{Z}_p \to N$ to be

$$h_{ab}(x) = g(f_{ab}(x)),$$

where $g : \mathbb{Z}_p \to N$ denotes the function

$$g(x) = x \qquad (\text{mod } n).$$

Then the family of hash functions H given by

$$H = \{h_{ab} | a, b \in \mathbb{Z}_p, a \neq 0\}$$

is a 2-universal family of hash functions.

Theorem 2 *H is a 2-universal collection of hash functions.*

Proof: For any given $x, y \in \mathbb{Z}_p, x \neq y$, we need to bound the number $\delta(x, y, H)$ of hash functions in H that map x and y to the same value.

We claim that this number is the same as the number $\delta(\mathbb{Z}_p, \mathbb{Z}_p, H)$ of pairs (r, s) in $\mathbb{Z}_p^2$ with $r \neq s$ such that $g(r) = g(s)$. This claim holds because in the *field* $\mathbb{Z}_p$ the system of equations

$$\begin{aligned} ax + b &= r \qquad (\text{mod } p) & (1)\\ ay + b &= s \qquad (\text{mod } p) & (2) \end{aligned}$$

has a unique solution (a, b) with $a \neq 0$ for any given $x, y, r, s \in \mathbb{Z}_p, x \neq y, r \neq s$. Thus, for every pair (r, s) there exists a unique function f_{ab} such that $(r, s) = (f_{ab}(x), f_{ab}(y))$. As there are only $p(p-1)$ such functions, we now know that $\{(f_{ab}(x), f_{ab}(y)) | a, b \in \mathbb{Z}_p, a \neq 0\} = \{(r, s) | r, s \in \mathbb{Z}_p, r \neq s\}$, and our claim follows from the definition of h_{ab} as $g \circ f_{ab}$.

It remains to compute $\delta(\mathbb{Z}_p, \mathbb{Z}_p, g)$. For any given $z \in N$, there are at most $\lceil \frac{p}{n} \rceil$ elements in $\mathbb{Z}_p$ that are mapped to z by g. Thus, any given element in Z_p collides with at most $\lceil \frac{p}{n} \rceil - 1$ other elements in $\mathbb{Z}_p$. And therefore

$$\delta(\mathbb{Z}_p, \mathbb{Z}_p, g) \leq p(\lceil \frac{p}{n} \rceil - 1) \leq \frac{p(p-1)}{n}.$$

Since there are $p(p-1)$ hash functions in H, it follows that H is 2-universal. ■

Note that all the functions in H can be computed efficiently and can be described using $2\lceil \log m \rceil$ bits.

3 Perfect Hashing

We have seen that in the dynamic setting we can achieve $O(1)$ *expected* running time for all the operations on the hash table. Can we do any better if we know the set S in advance? More specifically, given the set S, can we construct in reasonable time a hash table of size $O(s)$ such that all the operations take $O(1)$ *worst case* running time. For the sake of simplicity, we shall assume that $s = n$ in the following discussion.

Definition 3 *A family $H \subset \{h : \mathcal{U} \to N\}$ of hash functions is called* perfect*, if for every subset $S \subset \mathcal{U}$ there exists a hash function $h \in H$ such that h is perfect for S.*

A perfect collection of hash functions H would solve our problem, because given a set S, we could search this family until we find a function that does not cause any collisions in S. However, we want H to be small so that every function in H can be represented succinctly. In particular, we want $\log |H| \le c \log m$ for a fixed constant c, which means that $|H| \le m^c$.

Can we achieve this? Or, more generally, how large does a perfect hash family have to be? Using the probabilistic method, we can prove an upper bound.

Theorem 4 *There exists a family H of hash functions of size $O(e^s s \log m)$ such that H is perfect for every set of size at most s where m is the size of the universe.*

Proof: Fix a set $S \subset \mathcal{U}$ with $|S| = s$. Now, pick a random hash function $h : \mathcal{U} \to N$. We want to compute the probability that h is perfect for S. h is perfect if and only if $h_{|S}$ is bijective. There are s^s functions from S to N, $s!$ of which are bijective. Therefore, we get

$$\Pr[h \text{ is perfect for } S] = \frac{s!}{s^s} \sim \frac{1}{e^s},$$

where the last approximation is due to Stirling's approximation $n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$.

It follows that if we pick te^s hash functions, we get that

$$\Pr[\text{none of these functions is perfect for } S] = \left(1 - \frac{1}{e^s}\right)^{te^s} \sim e^{-t}.$$

There are $\binom{m}{s}$ subsets of size s of $\mathcal{U}$. Thus, if we pick t hash functions, the probability that there exists a set S of size s for that we picked no perfect hash function is at most $\binom{m}{s} e^{-t}$ by union bounds. If this probability is strictly less than one, we know that there exists a perfect hash family of size te^s. For that, we need $t \approx \log \binom{m}{s} \approx s \log \frac{m}{s}$. ■

Considering our requirement for the size of the perfect hash family, this is only feasible if $e^s = O(m^c)$ which means that $s = O(\log m)$. This is not very realistic. For example, for a universe of size $m = 2^{32}$, which corresponds to the most common word size today, we would only be able to store around 32 entries using a reasonably large perfect class of hash functions.

However, we only proved an upper bound. Maybe, there exists a perfect family of hash functions that is much smaller. Unfortunately, this is not the case.

Theorem 5 *Every perfect hash family for sets of size s contains at least $2^{\Omega(s)}$ elements.*

The proof is left as an exercise to the interested reader.

Perfect hash families do not seem to help us achieve our goal. However, two simple modifications to the scheme make give us what we need.

Firstly, if we relax the notion of perfect hash families to somewhat perfect hash families, we can get smaller families.

Definition 6 *A family $H \subset \{h : \mathcal{U} \to N\}$ of hash functions is called d-*perfect *for $d \geq 1$, if for every subset $S \subset \mathcal{U}$ there exists a hash function $h \in H$ such that for every $x \in S$ we have that $|\{y \in S \mid h(x) = h(y)\}| \leq d$, i. e. at most d elements are mapped to the same entry in the hash table.*

d-perfect hash families can be small.

Theorem 7 *There exists a $O(\log m)$-perfect family of hash functions of size at most m.*

Furthermore, if we allow our hash table to be larger than $O(s)$, we can also achieve smaller families of hash functions.

Theorem 8 *There exists a fixed constant c such that if $n \geq cs^2$, there exists a family of hash functions of size at most m that is perfect for sets of size at most s.*

Proof:

Once again, we use the probabilistic method. Fix a set S of size s and pick a random hash function $h : \mathcal{U} \to N$. We want to compute the probability that h is perfect for S. Thus, $h_{|S}$ must be injective. The number of injective functions from S to N is $n(n-1)\ldots(n-s+1)$. The number of all functions from S to N is n^s. Thus, we get

$$\begin{aligned}
\Pr[h \text{ is perfect for } S] &\leq \frac{n(n-1)\ldots(n-s+1)}{n^s} \\
&= 1\left(1-\frac{1}{n}\right)\ldots\left(1-\frac{s-1}{n}\right) \\
&\sim e^{-\frac{1}{n}-\frac{2}{n}-\ldots-\frac{s-1}{n}} \\
&\sim e^{-\frac{s^2}{n}}.
\end{aligned}$$

Consequently, if we pick m hash functions at random, the probability that there is a set S of size s for which none of the functions we picked is perfect is at most $\binom{m}{s}\left(1-e^{-\frac{s^2}{n}}\right)^m \approx \left(\frac{m}{s}\right)^s\left(1-e^{-\frac{s^2}{n}}\right)^m$. If this probability is strictly less than one, then we know a perfect hash family of size m exists. Therefore, n has to be larger than $\frac{s^2}{-\log\left(1-\left(\frac{s}{m}\right)^{\frac{s}{m}}\right)}$. Simple calculus shows that the coefficent of s^2 is bounded, and therefore there exists a c such that $n = cs^2$ is large enough. ■

Using these ideas, we can use the following scheme. Assume we have a family of hash functions H such that for every set $S \subset \mathcal{U}$ of size s, there is a $h \in H$ such that $\sum_{i=1}^{n} b_i^2 = O(s)$, where $b_i = |\{x \in \mathcal{U} \mid h(x) = i\}|$ denotes the number of elements that are mapped to entry i. Furthermore, assume that for every table size n we have a family of has functions H'_n such that for every set $S \subset \mathcal{U}$ such that $s^2 = n$, H'_n contains a hash function that is perfect for S. Then, given a set S, we can construct a data structure that provides $O(1)$ worst case time per Find operation by using to levels of hash tables. The first level hash table is of size s and is indexed using a hash function from H. Assume that b_i elements are mapped to the same location i. These elements are stored in a second level hash table of size b_i^2 using a hash function from $H'_{b_i^2}$ where different second level hash tables can have different hash functions. Given the properties of the hash families, we can choose hash functions such that the total size of the data structure is O(s) and such that there are no collisions in the second level hash tables.

There exists a simple family of hash functions with the above properties. Assume that $p = m+1$ is a prime number. Consider the set $V \subset M$ of numbers that represent the elements S and the set $R = \{0, 1, \ldots, r-1\}$ of indices for the hash table, where $r \geq |V|$. Define $h_k : M \to R$ as $h_k(x) = (kx \pmod{p}) \pmod{r}$. We claim that the family $H = \{h_k \mid 1 \leq k \leq p-1\}$ has the desired properties for appropriate values of r. The proof of this claim is left as an exercise to the reader.