Practice Problems: Hashing (Sample Solution)

Topics: Hash Tables.

Problem 3-1. [CLRS 11.4-4] Suppose that we use double hashing to resolve collisions–that is, the hash function is defined as $h(k,i) = (h_1(k) + i \cdot h_2(k)) \mod m$. Show that if m and $h_2(k)$ have a greatest common divisor $d \geq 1$ for some key k, then an unsuccessful search for key k examines $(1/d)$th of the hash table before returning to slot $h_1(k)$.

Soln. Fix k and let $h_1(k) = a$ and $h_2(k) = b$. Then the sequence $h(k,0), h(k,1), \ldots, h(k,m-1)$ each taken modulo m is the sequence $a, a+b, a+2b, \ldots a+(m-1)b$ taken $\mod m$. If m and b are not relatively prime, then the sequence $b, 2b, \ldots, (m-1)b$ repeats itself. To show this, let g be the greatest common divisor of b and m. Then, m divides $b \cdot (m/g)$. [This holds, since, $b = g \cdot b'$ and $m = g \cdot m'$ and so $b \cdot (m/g) = gb'm' = b'm$ and obviously b divides $b'm$. Hence m divides $b \cdot (m/g)$.] Hence, the sequence $0, b, 2b, \ldots, (m-1)/g$ repeats itself. If m divides $b \cdot (m/g')$ for any g' then, g' must be a common divisor of b and m. Since, g is the largest common divisor, m/g' is the period of repetition of the sequence $0, b, 2b, \ldots, (m-1)g$.

We can now apply this to the double hashing method for resolving collisions in open-addressing. If a key k is not in the hash table, and $h_1(k) = a$ and $h_2(k) = b$ say, then, the double hashing method checks only the slots $a, a+b, a+2b \ldots a+(m-1)/g$ all $\mod m$ and after that the sequence repeats itself. Hence, only $1/g$th fraction of the hash table is examined.

Problem 3-2. [CLRS 11-4] Let $\mathcal{H}$ be a class of hash functions in which each hash function $h \in \mathcal{H}$ maps the universe U of keys to $\{0, 1, \ldots, m-1\}$. We say that $\mathcal{H}$ is t-universal if, for any given sequence of t distinct keys, $k_1, k_2, \ldots, k_t$, and for any h chosen uniformly at random from $\mathcal{H}$, the sequence $h(k_1), h(k_2), \ldots, h(k_t)$ is equally likely to be any one of the m^t sequences of length t drawn from $\{0, 1, \ldots, m-1\}$.

a. Show that if $\mathcal{H}$ is a 2-universal family, then, $\mathcal{H}$ is universal.

Soln. Suppose $\mathcal{H}$ is 2-universal. Then, for distinct $k, l \in U$ and $a, b \in \{0, 1, \ldots, m-1\}$, we have,

$$\Pr_{h \in \mathcal{H}}[h(k) = a \text{ and } h(l) = b] = \frac{1}{m^2} \ .$$

Therefore,

$$
\begin{aligned}
\Pr_{h\in\mathcal{H}}[h(k)=h(l)] &= \sum_{a\in\{0,1,\ldots,m-1\}} \Pr_{h\in\mathcal{H}}[h(k)=a \text{ and } h(l)=b] \\
&= \sum_{a\in\{0,1,\ldots,m-1\}} \frac{1}{m^2} \\
&= \frac{m}{m^2} \\
&= \frac{1}{m}
\end{aligned}
$$

which satisfies the universality property.

b. Construct a specific family $\mathcal{H}$ that is universal, but not 2-universal, and justify your answer. Write down the family as a table, with one column per key, and one row per function. Try to make $m, \mathcal{H}$, and U as small as possible.

Soln.

Consider $U = \{a, b, c\}$, $\mathcal{H} = \{h_1, h_2, h_3, h_4\}$, where, each of the functions $h_1, \ldots, h_4$ map the domain $\{a, b, c\} \to \{0, 1\}$ and are defined as follows.

	a	b	c
h_1	0	0	0
h_2	0	0	1
h_3	1	0	0
h_4	1	0	1

Let us count the number of solutions for $h(k) = h(l)$, for distinct k and l.

1. Let $\{k, l\} = \{a, b\}$. Then, for $h \in \{h_1, h_2\}$, $h(a) = h(b)$ and for $h \in \{h_3, h_4\}$, $h(a) \neq h(b)$. Hence, there are 2 out of 4 hash functions satisfying $h(a) = h(b)$.
2. Let $\{k, l\} = \{b, c\}$. Then, for $h = h_1$ or h_3, $h(b) = h(c)$ and for $h = h_2$ or h_4, $h(a) \neq h(b)$. Hence there are 2 out of 4 hash functions satisfying $h(b) = h(c)$.
3. Lt $\{k, l\} = \{a, c\}$. Then, for $h = h_1$ or $h = h_4$, $h(a) = h(c)$ and for $h = h_2$ or h_3, $h(a) \neq h(c)$. Hence there are 2 out of 4 hash functions satisfying $h(a) = h(c)$.

The above calculations show that for any $k, l \in \{a, b, c\}$ and distinct, $\Pr_{h\in H}[h(k) = h(l)] = 2/4 = \frac{1}{2} = \frac{1}{m}$, since, $m = 2$. Thus, the family is universal.

However, the family is not 2-universal, since, there is no solution to $h(a) = 0$ and $h(b) = 1$. Hence, $\Pr_{h\in H}[h(a) = 0 \text{ and } h(b) = 1] = 0$, whereas for the family to be 2-universal, this probability should have been $1/m^2 = 1/4$.

c. Suppose that the universe U is the set of n-tuples of values drawn from $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, where, p is prime. Consider an element $x = (x_0, x_1, \ldots, x_{n-1}) \in U$. For any n-tuple $a = (a_0, a_1, \ldots, a_{n-1}) \in U$, define the hash function h_a by

$$h_a(x) = \left(\sum_{j=0}^{n-1} a_j x_j\right) \mod p \ .$$

Let $\mathcal{H} = \{h_a\}$. Show that $\mathcal{H}$ is universal but not two universal. (*Hint*: Find a key for which all hash functions in $\mathcal{H}$ produce the same value.)

Soln. Let x, y be two distinct n-tuples over U. Then, $h_a(x) = h_a(y)$ iff

$$\sum_{j=0}^{n-1} a_j x_j = \sum_{j=0}^{n-1} a_j y_j \mod p$$

or, equivalently,

$$\sum_{j=0}^{n-1} a_j (x_j - y_j) = 0 \mod p$$

Since, $x \neq y$, there exists at least one index s such that $x_s \neq y_s$. So the above equation can be written as

$$a_s(x_s - y_s) = - \sum_{\substack{0 \le j \le n-1 \\ j \neq s}} a_j (x_j - y_j) \mod p$$

Hence,

$$a_s = -(x_s - y_s)^{-1} \sum_{\substack{0 \le j \le n-1 \\ j \neq s}} a_j (x_j - y_j) \mod p$$

We can interpret the above equation as follows. Choose $a_0, \ldots, a_{s-1}, a_{s+1}, \ldots, a_{n-1}$ to be any values in $\mathbb{Z}_p$. Then, the value of a_s is fixed as per the above equation. Note that $(x_s - y_s)^{-1}$ is the unique multiplicative inverse of $x_s - y_s$ in $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$. Also note that $a = -b \mod p$ is a shorthand for saying that $a = (p - b) \mod p$, since, $-b$ means $p - b$. [More precisely, $-b$ refers to the additive inverse in $\mathbb{Z}_p$, that is, the unique number which when added to b gives $0 (\mod p)$. This number is obviously $p - b$].

Thus, the number of solutions to the equation $h_a(x) = h_a(y)$ is p^{n-1}. Why? since, all the a_j's except a_s may take any values in $\mathbb{Z}_p$, this determines the unique value of a_s. Thus, for $x \neq y$, $x, y \in \mathbb{Z}_p^n$,

$$\Pr_{a \in \mathbb{Z}_p^n} [h_a(x) = h_a(y)] = \frac{p^{n-1}}{p^n} = \frac{1}{p} \ .$$

The denominator p^n is the number of hash functions, since, each hash function corresponds with some $a \in \mathbb{Z}_p^n$. Since, $p = m$, the hash family is universal.

It is obviously not 2-universal, since, the all zeros vector $x = (0, 0, \ldots, 0)$ is mapped to 0 by all hash functions. Hence, for any non-zero vector,

$$\Pr_{a \in \mathbb{Z}_p^n} [h_a(0) = 1] = 0$$

and hence, the family is not 2-universal. (For it to be 2-universal, the above probability should have been $1/m = 1/p$.)

d. Suppose that we modify $\mathcal{H}$ slightly from part (**b**): for any $a \in U$ and for any $b \in \mathbb{Z}_p$, define

$$h_{a,b}(x) = \left(\sum_{j=0}^{n-1} a_j x_j + b \right) \mod p \ .$$

Let $\mathcal{H} = \{h_{a,b} \mid a \in U, b \in \mathbb{Z}_p\}$. Show that $\mathcal{H}$ is 2-universal.

Soln. Fix $\alpha, \beta \in \mathbb{Z}_p$ and $x, y \in \mathbb{Z}_p^n$ with $x \neq y$. We wish to count the number of solutions to the simultaneous equations $h_{a,b}(x) = \alpha$ and $h_{a,b}(y) = \beta$. This is equivalent to

$$b + \sum_{j=0}^{n-1} a_j x_j = \alpha \mod p \tag{1}$$

$$b + \sum_{j=0}^{n-1} a_j y_j = \beta \mod p \tag{2}$$

Since, $x \neq y$, there exists an index s such that $x_s \neq y_s$. Subtracting, the above equation implies that

$$a_s(x_s - y_s) = - \sum_{\substack{0 \le j \le n-1 \\ j \neq s}} a_j(x_j - y_j) + (\alpha - \beta) \mod p \tag{3}$$

or,

$$a_s = -(x_s - y_s)^{-1} \left(\sum_{\substack{0 \le j \le n-1 \\ j \neq s}} a_j(x_j - y_j) + (\alpha - \beta) \right) \mod p$$

Thus, for each choice of $a_1, \ldots, a_n$ except a_s, there is a unique choice of a_s from the above equation. Hence, the number of solutions to the equation (3) is p^{n-1}. Eqn. (3) can be written as

$$\sum_{j=0}^{n-1} a_j x_j - \alpha = \sum_{j=0}^{n-1} a_j y_j - \beta \mod p$$

The simultaneous equations in (1) and (2) are equivalent to the following simultaneous equations.

$$\sum_{j=0}^{n-1} a_j x_j - \alpha = \sum_{j=0}^{n-1} a_j y_j - \beta \qquad \mod p \tag{4}$$

$$b = - \sum_{j=0}^{n-1} a_j x_j + \alpha \qquad \mod p \tag{5}$$

The number of solutions in terms of the n tuple $(a_0, a_1, \ldots, a_{n-1})$ for (4) is calculated as p^{n-1}. For each such solution, there is a unique solution for b for (5). Hence, the total number of solutions of $n+1$-tuple $(a_0, a_1, \ldots, a_{n-1}, b)$ for equations (4) and (5) are p^{n-1}.

Hence, for $x, y \in \mathbb{Z}_p^n$ and distinct and $\alpha, \beta \in \mathbb{Z}_p$,

$$\Pr_{a \in \mathbb{Z}_p^n, b \in \mathbb{Z}_p} [h_{a,b}(x) = \alpha \text{ and } h_{a,b}(y) = \beta] = \frac{p^{n-1}}{p^{n+1}} = \frac{1}{p^2} \ .$$

Hence the family is 2-universal.

e. *Application to authentication.* Suppose that *Ranbir* and *Katrina* secretly agree on a hash function h from a 2-universal family $\mathcal{H}$ of hash functions, where, each $h \in H$ maps the universe of keys U to $\mathbb{Z}_p$, where, p is prime. Now *Ranbir* sends a message m over the internet to *Katrina* and *authenticates* this message by also sending a tag $t = h(m)$. *Katrina* receives the pair (m, t) and verifies that indeed $h(m) = t$. If the verification succeeds, then she accepts the message, and otherwise discards it. However, there could be many a snooping *Mr. Mediaman* on the internet who can intercept the message (m, t) and replace it with (m', t') and deliver it to *Katrina*. Suppose that the snooping *Medianman* knows the hash family $\mathcal{H}$ (but not the choice h agreed upon by *Ranbir* and *Katrina*). Show that the probability with which *Mr. Mediaman* may succeed in fooling *Katrina* is at most $1/p$, irrespective of how much computing power the *Mediaman* has.

Soln. Let $h_0 \in \mathcal{H}$ be the hash function secretly agreed upon by *Ranbir* and *Katrina*. The receiver *Katrina* upon receiving m', t' checks if $h_0(m') = t'$. The *Mediaman* can fool *Katrina* iff $h_0(m') = t'$. But *Mediaman* does not know h_0. So, how many hash functions map m' to the same value as $h_0(m')$. That is, how many $h \in \mathcal{H}$ satisfy the equation

$$h(m') = h_0(m')$$

Since, h is 2-universal, and choosing any $y \in U$ such that $y \neq x$,

$$\Pr_h [h(x) = a] = \sum_{b \in \mathbb{Z}_p} \Pr_h [h(x) = a \text{ and } h(y) = b] = \sum_{b \in \mathbb{Z}_p} \frac{1}{p^2} = \frac{1}{p} \ .$$

Hence, the number of hash functions that satisfy $h(m') = h_0(m')$ is $1/p$th of the number of hash functions of U. Since, *Mediaman* does not know h_0, it can do no better than guess $h \in H$. This means the probability that the receiver *Katrina* is fooled is bounded by $1/p$.

Can the *Mediaman* do better? It receives the pair m, t and it can find a hash function h such that $h(m) = t$. But since $\mathcal{H}$ is universal, the number of hash functions such that $h(m) = h_0(m)$ is $1/p$ fraction of $\mathcal{H}$. Thus, the *Mediaman* cannot do better than reducing its choices by a factor of p.

Remark. Suppose that the *Mediaman* can observe not one, but k messages between the sender and the receiver. Then, it can form the simultaneous equations

$$\begin{aligned} h(m_1) &= t_1 \\ h(m_2) &= t_2 \\ &\vdots\vdots \\ h(m_k) &= t_m \end{aligned}$$

Finding all the hash functions satisfying these equations may be a smaller set and can seriously increase the chances of a successful impersonation.

Problem 3-3. [***Rolling Hash Functions for Pattern Matching***] You are given a large piece of text in the string of characters $T[1 \ldots n]$. Given a (significantly shorter) pattern $P[1 \ldots m]$, *the problem is to determine whether* P *occurs in* T, that is, if there exists some shift position $0 \leq s \leq n - m + 1$ such that $P[j] = T[s + j]$, for each $j = 1, \ldots, m$. Assume that each character is drawn from an alphabet of 64 characters (to allow upper-case and lower case characters and digits). We can represent any m-character string $C[0, \ldots, m-1]$ uniquely as a (large) integer $N(C[0 \ldots m-1])$ as follows:

$$N(C[0 \ldots m-1]) = C[0] + C[1] \cdot 2^6 + C[2] \cdot 2^{12} + \ldots + C[m-1] \cdot 2^{6 \cdot (m-1)}$$

a. Let h be a hash function that maps m character strings to $\mathbb{Z}_p$ (where p is a large prime). Further suppose that h is perfect, that is, for $x \neq y$, $h(x) \neq h(y)$. Assume that you can calculate the hash value of m-character strings in time $O(m)$. Design an algorithm that given the text string $T[1 \ldots n]$ and the pattern string $P[1 \ldots m]$, returns all positions in T where P occurs, in worst-case time $O(mn)$.

Soln. The idea is to hash each m-length substring $T[i \ldots i+m-1]$ and let $U[i] = h(T[i \ldots i+m-1])$. Also hash the m-length pattern $h(P)$. Then, P occurs in $T[1 \ldots m]$ starting at position i only if it $h(P) = U[i]$. Since it is given that the hash function is perfect, hence, $h(P) = U[i]$ implies that $P = T[i \ldots i+m-1]$. Thus, an algorithm can be designed as follows.

1. For $i = 1, 2, \ldots n - m + 1$ compute $U[i] = h(T[i \ldots i + m - 1])$.
2. Compute $h(P)$.
3. Find all i such that $h(P) = U[i]$, these are the starting positions of the occurrences of P in U.

Step 1 requires $O(mn)$ time, since, computing each $U[i]$ requires the computation of a hash function on an m-length string. This is given to take $O(m)$ time. Step 2 requires $O(m)$ time. Step 3 requires n comparisons with $h(P)$, and can be done in $O(mn)$ time. Thus, total time required is $O(mn + m + mn) = O(mn)$.

b. Suppose h is not necessarily perfect. Extend the previous algorithm to return all positions in T where P occurs, in worst-case time $O(mn)$.

Soln. We add a verification step. Step 3 identifies all positions i such that $h(P) = U[i]$. This guarantees that if $P = T[i \ldots i + m - 1]$ then $h(P) = h(T[i \ldots i + m - 1]) = U[i]$. But since the hash function is not perfect, it is possible that there are false positives, that is, $P \neq T[i \ldots i + m - 1]$ but $h(P) = h(T[i \ldots i + m - 1]) = U[i]$. Hence, after identifying a possible superset of the matching indices $\{i\}$, we verify that actually $P = T[i \ldots i + m - 1]$.

3' . For $i = 1, \ldots, n - m + 1$, if $h(P) = U[i]$, then test if $P[j] = T[i + j - 1]$, for each $j = 1, 2, \ldots, m$.

The verification step takes $O(m)$ time and is applied at most $n - m + 1$ times, for a total of $O(mn)$ time.

c. Fix a prime p and define the hash function

$$h_p(x) = x \mod p \ .$$

This hash function can be used to hash any m-character string $A[i \ldots i+m-1]$ by first converting it into an equivalent large number $N(A[i \ldots i+m-1])$ (as shown above) and then calculating

$$N(A[i \ldots i+m-1]) \mod p \ .$$

Show how to calculate the hash value of the string $A[(i+1)\ldots(i+m)]$ in $O(1)$ time if the hash value corresponding to the string $A[i \ldots (i+m-1)]$ has already been computed and its value is known.

Soln. From the definition of $N(\cdot)$,

$$\begin{aligned}
N(A[i \ldots m-1]) &= A[i] + A[i+1]\cdot 2^6 + A[i+2]\cdot 2^{12} + \ldots + A[i+m-1]\cdot 2^{6\cdot(m-1)} \\
N(A[i+1 \ldots m]) &= A[i+1] + A[i+2]\cdot 2^6 + A[i+3]\cdot 2^{12} + \ldots + A[i+m]\cdot 2^{6\cdot(m-1)} \\
&= (N(A[i \ldots m-1]) - A[i])/2^6 + A[i+m]\cdot 2^{6\cdot(m-1)}
\end{aligned}$$

Therefore,

$$\begin{aligned}
h_p(N(A[i+1 \ldots m]) &= h_p\left((N(A[i \ldots m-1]) - A[i])/2^6 + A[i+m]\cdot 2^{6\cdot(m-1)}\right) \\
&= (N(A[i \ldots m-1]) - A[i])(2^6)^{-1} + A[i+m]\cdot 2^{6\cdot(m-1)} \mod p \\
&= (h_p(N(A[i \ldots m-1])(2^6)^{-1} - A[i](2^{-6})^{-1} + A[i+m]\cdot 2^{6\cdot(m-1)} \mod p
\end{aligned}$$

We can pre-compute $(2^6)^{-1} \mod p$ and $2^{6\cdot(m-1)} \mod p$ as these are independent of the string index i. Hence, $h_p(N(A[i+1 \ldots m])$ can be computed using the formula above using 2 multiplications and 2 additions/subtractions $\mod p$. This can be done in $O(1)$ time.

This incremental computation of the hash value of $A[i+1,\ldots,i+m]$ from the previously computed hash value for $A[i \ldots i+m-1]$ gives the name *Rolling hash function*.

d. Let p be a prime in the range $[2, cn^3]$ for some positive constant c. Let $h_p(x) = x \mod p$ and let $\mathcal{H}$ be the family of hash functions $\mathcal{H} = \{h_p \mid p \text{ is prime and } 2 \le p \le cn^{24}\}$. Let P be the given pattern string of m characters and let $T[i \ldots i+m-1]$ be any m-length substring of T. Then, show that

$$\Pr_p\left[h_p(N(P)) = h_p(N(T[i \ldots i+m-1]))\right] \le \frac{1}{n}$$

holds for an appropriate choice of c. ***Hint:*** You could use the following two number theoretic facts: (1) an integer x has at most $\log x$ prime factors, and, (2) the *Prime Number Theorem*: there are $\Theta(x/\log(x))$ prime numbers in the range $[2, x]$.

Soln. Let $Q = T[i \ldots i+m-1]$. Let $a = N(P)$ and $b = N(Q)$. Suppose $a \ne b$ (otherwise, $h_p(a) = h_p(b)$ for all p). Now, $0 \le a, b, < 2^{6m}$.
Suppose $h_p(a) = h_p(b)$. Then, $a - b = 0 \mod p$ or p divides $a - b$. So, $a - b$ has at most $\log_2 |a-b| < 6m$ prime factors.
The number of prime numbers in the range $[2, cn^d]$ is, by the prime number theorem, $\Theta(cn^d/\log(cn^d))$.

Hence,

$$\Pr_{p\in[2,cn^d]}\left[h_p(a) = h_p(b)\right] \le \Theta\left(\frac{6m}{cn^d/\log(cn^d)}\right) = \Theta\left(\frac{6m\log(cn^d)}{cn^d}\right) \le \frac{1}{n}$$

assuming $d \ge 3$ (since, $m \le n$) and choosing c appropriately.