Perfect Hashing via Universal Hash Functions

Alon Itai

April 5, 2000

1 Introduction

Let $A = \{0, \ldots, a-1\}$ and $B = \{0, \ldots, b-1\}$. Let $S \subset A$ of size $n \leq b$. A function $h : A \to B$ is *perfect for* S if h restricted to S is one-one, i.e., for all $x \neq y \in S$, $h(x) \neq h(y)$.

Since $|S| \leq |B|$ there surely exists such a function. To find one we could search among all the a^b functions from A to B.

One such function is that that sends the minimum member of S to 0, the next element to 1, etc. i.e,. $h(x) = |\{y \in S | y < x\}|$. To compute $h(x)$, simply access the x'th position of this list. If the list is represented as a linear array then this can be accomplished in $O(1)$ time. Note, however that $O(a)$ space is required.

Alternatively, since we are interested only on the value of h on S, we could keep a list of pairs $\langle s, h(s)\rangle$, $(s \in S)$. The length of the list is $|S| = n$, and if it is sorted by s and kept in an array, the computation of h requires $O(\log n)$ time.

Fredman, Kolmós and Szemerédi [2] showed how to find a perfect hash function which requires $O(1)$ time to compute and occupies $O(s \log_2 b)$ space. This construction, known as FKS, requires one to choose functions from a set P of functions over finite fields. Carter and Wegman showed that a similar set of functions is universal (see Section 2). We will show that FKS's general construction works even if the specific functions are chosen from *any* universal class of hash functions, not just P. Our results, therefore, decouple FKS's construction from the properties of finite fields.

2 Universal Hash Functions

Following Carter and Wegman [3] (see also [1]) a set of functions $H \subset B^A$ is *universal* if for every pair of distinct keys $x, y \in A$, the number of functions $h \in H$ for which $h(x) = h(y)$ is exactly $1/b$. In other words, with a hash function randomly chosen from H, the chance of a collision between x and y with $x \neq y$ is exactly $1/b$. We call the set H a *universal family of hash functions (UFHF)*.

We follow Carter and Wegman's notation: Let A to B and S be as above and h be a hash function from A to B. Define

$$\delta_h(x, y) = \begin{cases} 1 & x \neq y \text{ and } h(x) = h(y), \\ 0 & \text{otherwise.} \end{cases}$$

We also follow their convention that if a set appears instead of either x, y or h, then we sum over the elements of that set. Thus, for example,

$$\delta_H(x, S) = \sum_{h \in H} \sum_{y \in S} \delta_h(x, y) \ .$$

By this convention, H is universal if and only if

$$\delta_H(x,y) = \frac{|H|}{|B|} \ . \tag{1}$$

Theorem 2.1 *Let $A = \{0,\ldots,a-1\}$ and $B = \{0,\ldots,b-1\}$. Then there exists a family H of universal hash functions from A to B, such that H contains at most $|A|^2$ functions.*

Proof: See family H_1 of [3]

□

3 The construction

Intuitively, if S is kept fixed and B increases the chances that H contains a perfect hash function should increase. The following lemma shows that this indeed holds when $|B| \geq n^2$.

Lemma 3.1 *Let H be a UFHF from A to B and $S \subseteq A$, such that $|S| = n$ and $|B| = n^2$. Then there exists $h \in H$ which is perfect for S.*

Proof: We show that there exists $h \in H$ for which $\delta_h(S,S) = 0$.

Since H is universal, equation (1) states:

$$\delta_H(x,y) \leq \frac{|H|}{|B|} \ .$$

Summing over all pairs

$$\begin{aligned} \delta_H(S,S) &= \sum_{x\in S}\sum_{y\in S}\delta_H(x,y) = \sum_{x\in S}\sum_{y\in S\setminus\{x\}}\delta_H(x,y) = \sum_{x\in S}\sum_{y\in S\setminus\{x\}}\frac{|H|}{|B|} \\ &= |S|\,(|S|-1)\,\frac{|H|}{|B|} \ . \end{aligned} \tag{2}$$

Since $|S| = n$ and $|B| = n^2$,

$$\delta_H(S,S) < |H| \ .$$

Hence,

$$E_h(S,S) = \frac{\delta_H(S,S)}{|H|} < 1 \ .$$

Since the minimum is less than or equal to the average, and $\delta_h(S,S)$ is integer, for some $h \in H$

$$\delta_h(S,S) = 0 \ .$$

□

We can construct a family H of functions, each of which can be computed in $O(1)$ time, thus fulfilling the time demands. However, in most cases, it is impractical to keep a hash table of size n^2, so we are not done yet. We follow Fredman et al. and maintain an auxiliary hash table C of size n and initially map S to C using a hash function h', not necessarily perfect for S. Let $S_i = \{s \in S : h'(s) = i\}$ and $\ell_i(h') = |S_i|$. We map S_i to a hash table B_i of size $[\ell_i(h')]^2$, using a hash function h_i perfect for S_i. The function h_i is chosen from a universal family of hash functions H_i from A to $0..[\ell_i(h)]^2 - 1$. (By Lemma 3.1 H_i contains a hash function perfect for S_i.)

Let $m_i = \sum_{j<i} [\ell_j(h')]^2$. The final hash table is the concatenation of these tables. Thus its size is $\sum_{i=1}^{n} [\ell_i(h')]^2$.

In the i'th entry of the auxiliary table C we store m_i and h_i. To compute $h(x)$ we first compute $i = h'(x)$ and then $h(x) = m_i + h_i(x)$. If each hash function is computed in $O(1)$ time, h can be also computed in $O(1)$ time.

In order to maintain linear space, we have to show that we can choose h' such that the size of the final table, $\sum_{i=1}^{n} [\ell_i(h')]^2$, is linear.

Lemma 3.2 *Let H be a UFHF from A to C, $S \subseteq A$, $|S| = |C| = n$. There exists $h' \in H$ such that*

$$\sum_{i=1}^{n} [\ell_i(h')]^2 \leq 2n \ .$$

Proof: Let $g \in H$ and let y satisfy $g(y) = i$. Since by definition $\delta_g(x,x) = 0$,

$$\sum_{x} \delta_g(x,y) = \ell_i(g) - 1.$$

$$\sum_{\substack{y \in S \\ g(y) = i}} \sum_{x} \delta_g(x,y) = \ell_i(g)(\ell_i(g) - 1) \ .$$

$$\begin{aligned} \sum_{y \in S} \sum_{x} \delta_g(x,y) &= \sum_{i=1}^{n} \sum_{\{y \in S,\ g(y)=i\}} \sum_{x} \delta_g(x,y) = \sum_{i=1}^{n} \ell_i(g)(\ell_i(g) - 1) \\ &= \sum_{i=1}^{n} [\ell_i(g)]^2 - \sum_{i=1}^{n} \ell_i(g) = \sum_{i=1}^{n} [\ell_i(g)]^2 - n \ . \end{aligned}$$

In other words,

$$\sum_{i=1}^{n} [\ell_i(g)]^2 = n + \delta_g(S,S) \ . \tag{3}$$

Since H is a UFHF, and by assumption $|C| = n$,

$$\delta_H(x,y) \leq \frac{|H|}{|C|} = \frac{|H|}{n} \ . \tag{4}$$

$$\delta_H(S,S) \leq \frac{|H|}{n} n^2 = |H| n \ .$$

The expectation (over H):

$$E_{g \in H} \delta_g(S,S) \leq n \ . \tag{5}$$

Since the minimum is less than or equal to the expectation, there exists $h' \in H$ such that $\delta_{h'}(S,S) \leq n$. Thus by (3),

$$\sum_{i=1}^{n} [\ell(h')]^2 = n + \delta_{h'}(S,S) \leq 2n \ .$$

□

4 Finding h' and the h_i's

Given sets A, C, $S \subseteq A$, and H a UFHF from A to C, one can check in time linear in $n = |S|$ if a function $h \in H$ satisfies Lemma 3.2. The problem is that H may contain up to $|A|^{|B|} = a^{2n}$ functions. However, by relaxing the constants of Lemma 3.2 we can show that half the functions of H satisfy the relaxed lemma. If we choose functions at random from H, the expected number of trials until finding a function that satisfies Lemma 3.2 is 2.

Lemma 4.1 *Let H be a UFHF from A to C, $S \subseteq A$, $|S| = n$, and $|C| = 2n$. At least half of the functions $h' \in H$ satisfy*

$$\sum_{i=1}^{n} [\ell_i(h')]^2 \leq 2n \ .$$

Proof: Following the proof of Lemma 3.2 equation (4) becomes now

$$\delta_H(x,y) \leq \frac{|H|}{|C|} = \frac{|H|}{2n} \ .$$

$$E_{g \in H} \delta_g(S,S) \leq \frac{n}{2} \ .$$

Hence for at least half the functions $h' \in H$, $\delta_{h'}(S,S) \leq n$.

$$\sum_{i=1}^{n} [\ell(h')]^2 = n + \delta_{h'}(S,S) \leq 2n \ .$$

□

It remains to show how to efficiently find perfect hash functions $h_i : S_i \to B_i$. We reuse the same trick, increase the size of B_i by a factor of 2, to get a target set B_i', and choose h_i at random from a UFHF from S_i to B_i'.

Lemma 4.2 *Let H_i be a UFHF from A to B_i' and $S_i \subseteq A$, $S_i \subseteq A$ such that $|S_i| = \ell_i$ and $|B_i'| = 2\ell_i^2$. Then at least half the functions of H_i are perfect for S_i.*

Proof: Substituting in equation (2)

$$\delta_{H_i}(S_i,S_i) = |S_i|\,(|S_i|-1)\,\frac{|H_i|}{|B_i'|} \ .$$

Since $|S_i| = \ell_i$ and $|B_i'| = 2\ell_i^2$,

$$\delta_{H_i}(S_i,S_i) < |H_i|/2 \ .$$

Since $\delta_{H_i}(S_i,S_i)$ is a non-negative integer, at least half the functions $h_i \in H_i$ satisfy $\delta_{h_i}(S_i,S_i) = 0$, i.e., are perfect for S_i. □

Thus the size of the final table, B,

$$\sum_{i=1}^{2n} 2\,|B_i'|^2 = \sum_{i=1}^{2n} 2\ell_i^2 = 4n \ .$$

To summarize: In order to find a perfect hash function in linear expected time we have increased the size of the auxiliary table C from n to $2n$, and the size of the final table B from $2n$ to $4n$.

Theorem 4.3 *Let $S \subseteq A$ be a set of size n and B a set of size $4n$. There exists a perfect hash function of size $2n$, and whose computation time is $O(1)$. The perfect hash function can be constructed in linear expected time from any UFHF.*

Remark 4.4: The constants 4 and 2 in the above theorem can be replaced by $2(1+\varepsilon)$ and $1+\varepsilon$ respectively, at the cost of increasing the running time by a factor of $O(1/\varepsilon)$.

5 Dynamic Perfect Hashing

We now wish to extend our results to allow insertions and deletion into the set S. Assume first that only insertions are conducted.

The dynamic structure is conducted in phases. At the beginnig of the p'th phase there are $n = 2^p$ elements in the hash table. and at its end there are $2^{n+1} - 1 = 2n - 1$.

Choose h' such that

$$\sum \ell_i^2 \leq 4n \ ,$$

and map S_i to a subtable of size

$$r_i = 2(2\ell_i)^2 = 8\ell_i^2 \ .$$

Allocate for B a table of size Kn (K is a constant whose value will be determined later).

Inserting x

1. If the phase has ended, reorganize the table to a table twice the size and choose new h' and h_i's.
2. $i = h'(x)$
   ℓ_i++.
3. If $\sum \ell_i^2 > 4n$, choose a new h'.
4. If $2\ell_i^2 \geq r_i$, reallocate the i-th subtable of size $4r_i$ in a new location, and choose a new h_i.
5. If location $h_i(x)$ in the i-th subtable is occupied, choose a new h_i and reorganize the i-th subtable (in situ).
6. Add x at the $h_i(x)$ position of the i-th subtable.

Lemma 5.1 *If $K \geq 42\frac{2}{3}$ then the table B does not overflow.*

Proof: By step 3, throughout the algorithm

$$\sum \ell_i^2 \leq 4n \ .$$

Let $r_i^{(F)}$ be the last value of r_i.

$$\begin{aligned} r_i^{(F-1)} &= \frac{r_i^{(F)}}{4} \\ r_i^{(F-2)} &= \frac{r_i^{(F)}}{16} \\ &\ldots \end{aligned}$$

$$\sum_j r_i^{(j)} = r_i^{(F)} + \frac{1}{4} r_i^{(F)} + \frac{1}{16} r_i^{(F)} + \ldots = \frac{4}{3} r_i^{(F)} \ .$$

$$|B| = \sum_i \sum_j r_i^{(j)} \leq \frac{4}{3} \sum_i r_i^{(F)} \leq \frac{4}{3} \sum_i 8\ell_i^2 \leq \frac{64}{3} 4n = \frac{128}{3} n \ .$$

$\square$

Deletions:

If a hash function is perfect for S it is also perfect for $S' \subseteq S$. The problem however is that the spaced used might be proporitonal to the total number of elements inserted, not the number of elements currently in the table. However, on reorganization of the entire table or subtables, we need to pick functions that satisfy the requrements for the current S, not including the deleted elements. Moreover, we can initiate additional reorganizations whenever the number of deletions since the last reorganization exceeds the number of undeleted elements in the table. This increases the time by a constant, but the space is proportional to the size of the current set.

References

[1] Corman, Leiserson, and Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[2] Fredman, Kolmós, and Szemerédi. Storing a sparse table with $o(1)$ worse case access time. *J. ACM*, 31:538–544, 1984.

[3] C. J. Lawrence and W. M. N. Universal classes of hash functions. *JCSS*, 18:143–154, 1979.