IEEE Symposium on Foundations
of Computer Science 82
6

On the Program Size of Perfect and Universal Hashfunctions

(Extended Abstract)

by

Kurt Mehlhorn
Fachbereich 1o
Universität des Saarlandes
66oo Saarbrücken
West - Germany

Abstract: We address the question of program size of perfect and universal hash functions. We prove matching upper and lower bounds (up to constant factors) on program size. Furthermore, we show that minimum or nearly minimum size programs can be found efficiently. In addition, these (near) minimum size programs have time complexity at most O(log* N) where N is the size of the universe.

Hashing is a very popular solution for the dictionary problem, i.e. for the problem of maintaining a set under operations Access, Insert and Delete. We use the following notation throughout this paper

$U = [0...N-1]$	is the universe
$S \subseteq U$	is the set stored
$n = \|S\|$	is the size of S
$T[0...m-1]$	is the hash table,
m	is the table size
$\beta = n/m$	is the load factor

Hashing is an O(1) average case solution for the dictionary problem. Unfortunately, its worst case is O(n). There have been two attacks on the worst case behaviour of hashing recently: Perfect Hashing (Sprugnoli , Jaeschke, Tarjan/Yao) and Universal Hashing (Carter/ Wegman).

Perfect Hashing only deals with static sets, i.e. only operation Access is supported.

Definition: a) A function $h : [0...N-1] \to [0...m-1]$ is a perfect hash function for $S \subseteq [0...N-1]$ if $h(x) \neq h(y)$ for all $x,y \in S$, $x \neq y$.

b) A set H of functions $h : [0...N-1] \to [0...m-1]$ is called (N,m,n) - perfect (or perfect for short) if for every $S \subseteq [0...N-1]$, $|S| = n$, there is $h \in H$ such that h is perfect for S. □

Sprugnoli and Jaeschke discuss henristic methods for finding perfect hash functions. The only theoretical paper in the area is by Tarjan/ Yao. They show that for $m = n$, $N = n^k$ for some fixed k, and any $S \subseteq U$, $|S| = n$, there is always a program of length at most O(n log n) and time complexity O(1) in the unit cost measure which computes a perfect hash function for S. The most interesting questions concerning perfect hash functions are:

a) How large are the programs for perfect hash functions, i.e. is there always a short program computing a perfect hash function?

b) How hard is it to find a perfect hash function given S, m and N.

c) How hard is it to evaluate perfect hash functions?

Program size is defined as follows. We assume that programs are written in some fixed programming language, say PASCAL. Then program size is the length of the program as a string over some finite alphabet. We prove upper bounds on program size by explicitly exhibiting programs of a certain length. We prove lower bounds on program length by deriving a lower bound on the number of different programs required and then using the fact that the number of words (and hence programs) of length $\leq L$ over an alphabet of size c is $\leq c^{L+1}/(c-1)$.

Theorem A: For every $S \subseteq U$, $|S| = n$, there is a program of length $O(\beta n + \log\log N)$ which computes a perfect hashfunction for S and for most $S \subseteq U$, $|S| = n$, the shortest program which computes a perfect hash function for S has size $\Omega(\beta n + \log\log N)$

Remark: Theorem A settles question a) completely. It has an interesting interpretation. The program size of perfect hash functions consists of an initial cost of $O(\log\log N)$ due to the size of the universe and an incremental cost of $O(\beta)$ per element stored.

Proof of theorem A: The proof of the lower bound is based on the following lemma.

Lemma 1: Let $N,m,n \in \mathbb{N}$ and let H be a (N,m,n) - perfect class of hash functions. Then

a) $|H| \geq \binom{N}{n}/((N/m)^n \cdot \binom{m}{n})$

b) $|H| \geq \log N/\log m$

Proof: a) There are $\binom{N}{n}$ different subsets of $[0\ldots N-1]$ of size n. It therefore suffices to show that any fixed function $h : [0\ldots N-1] \to [0\ldots m-1]$ is perfect for at most $(N/m)^n \cdot \binom{m}{n}$ different subsets $S \subseteq [0\ldots N-1]$, $|S| = n$. If h is perfect for S then $|h^{-1}(i) \cap S| \leq 1$ for all $i \in [0\ldots m-1]$. Hence the number of sets S such that h is perfect for S is bounded by

$$\sum_{0 \leq i_1 < i_2 < \ldots < i_n < m} |h^{-1}(i_1)| \cdot |h^{-1}(i_2)| \ldots |h^{-1}(i_n)|$$

This expression is maximal if $|h^{-1}(i)| = N/m$ for all $i \in [0\ldots m-1]$ and its value is equal to $(N/m)^n \cdot (m/n)$ in this case.

b) Let $H = \{h_1, \ldots, h_t\}$. We construct $U_i \subseteq U$, $0 \leq i \leq t$, such that for every $S \subseteq U$, $|S \cap U_i| \geq 2$, functions $h_1, \ldots, h_i$ are not perfect for S. Then we must have $|U_t| \leq 1$. Let $U_0 = U$ and

$$U_{i+1} = U_i \cap h_{i+1}^{-1}(j) \quad \text{for } i < t$$

where j is such that $|U_i \cap h_{i+1}^{-1}(j)| \geq |U_i \cap h_{i+1}^{-1}(\ell)|$ for every $\ell \in [0\ldots m-1]$. Then $|U_{i+1}| \geq |U_i|/m$ and hence $|U_{i+1}| \geq N/m^{i+1}$. Also functions $h_1, \ldots, h_{i+1}$ are constant on U_{i+1} and hence $|U_t| \leq 1$. Thus $1 \geq N/m^t$ or $t \geq \log N/\log m$. □

The proof of the lower bound is now completed by taking logarithms on both sides of the inequalities derived in lemma 1. The upper bound on program size is derived in a two step process. In the first step we show by counting that there is always a perfect class of small size, in the second step we show that step 1 can be done by small programs.

Lemma 2: Let $N,m,n \in \mathbb{N}$. If

$$t \geq n \cdot \ell n\, N \cdot e^{n^2/m}$$

then there is a (N,m,N) - perfect class H with $|H| = t$.

Proof: We may represent any class $H = \{h_1, \ldots, h_t\}$ by a N by t matrix $M(H) = (h_i(x))_{0 \le x \le N-1, 1 \le i \le t}$ with entries in [0...m-1], i.e. the i-th column of matrix M(H) is the table of function values for h_i. Conversely, every N by t matrix M is the representation of a class H of hash functions. There are $m^{N \cdot t}$ matrices of dimension N by t with entries in [0...m-1].

We want to derive an upper bound on the number of non-perfect matrices, i.e. matrices which correspond to non-perfect classes of hash functions. If H does not contain a perfect hash function for $S = \{x_1 < x_2 < \ldots < x_n\}$, then the submatrix of M(H) given by rows $x_1, x_2, \ldots, x_n$ cannot have a column with n different values. Hence the columns of that submatrix can be chosen out of $m^n - m(m-1)\ldots(m-n+1)$ possibilities (namely, the number of functions from n points into a set of m elements minus the number of injective functions), and hence the number of such submatrices is bounded by $[m^n - m(m-1)\ldots(m-n+1)]^t$. Recall that the submatrix has t columns. Since S can be chosen in $\binom{N}{n}$ different ways, the number of non-perfect matrices is bounded by

$$\binom{N}{n} [m^n - m(m-1)\ldots(m-n+1)]^t \; m^{(N-n)\cdot t}$$

Note that the rows corresponding to elements not in S may be filled arbitrarily. Thus there is a perfect class H, $|H| = t$, if

$$\binom{N}{n} [m^n - m(m-1)\ldots(m-n+1)]^t \; m^{(N-n)\cdot t} < m^{N \cdot t}$$

A short calculation shows that this inequality certainly holds for $t \ge n(\ln N)\, e^{n^2/m}$. □

Lemma 2 gives us an upper bound on the cardinality of perfect classes of hash functions; it does not yet give us an upper bound on program size. The upper bound on program size is given by the next lemma.

Lemma 3: Let $N, m, n \in \mathbb{N}$. For every $S \subseteq [0\ldots N-1]$, $|S| = n$, there is a program of length

$$O(n^2/n + \log\log N + 1)$$

which computes a perfect hash function $h : [0\ldots N-1] \to [0\ldots m-1]$ for S.

Proof: We will explicitely describe a program. The program implements the proof of lemma 6; it has essential 4 lines:

(1) $k \leftarrow \lceil \ell n\, N \rceil$ written in binary;

(2) $t \leftarrow \lceil n \cdot k \cdot e^{n^2/m} \rceil$ written in binary;

(3) i ← some number between 1 and t depending on S written in binary;

(4) search through all 2^k by t matrices with entries in [0...m-1] until a (N,m,n) - perfect matrix is found; use the i-th column of that matrix as the table of the hash function for S;

The correctness of this program is immediate from lemma 2; by lemma 2 there is a 2^k by t perfect matrix and hence we will find it in step (4). One column of that matrix describes a perfect hash function for S, say the i-th. We set i to the appropriate value in line (3).

The length of this program is log log N + O(1) for line (1), log n + log log N + n^2/m + O(1) for lines (2) and (3) and O(1) for line (4). Note that the length of the text for line (4) is independent of N, t and m. This proves the claim. □

Theorem A characterizes the program size of perfect hash functions; upper and lower bounds are of the same order of magnitude. Unfortunately the program constructed in lemma 3 is completely useless. It runs extremely slow and it uses an immense amount of work space. Therefore, we have to look for constructive upper bounds if we also want to get some insight on questions b) and c).

An answer to b) and c) for very small load factor is given by

Theorem B: Let $S = \{x_1 < x_2 < \ldots < x_n\} \subseteq [0..N-1]$.

a) There is a prime $p = O(n^2 \ln N)$ such that $x_i \bmod p \neq x_j \bmod p$ for $i \neq j$.

b) A number p satisfying $p = O(n^2 \ln N)$ and $x_i \bmod p \neq x_j \bmod p$ for $i \neq j$ can be found in time $O(n \cdot \log n \cdot (\log n + \log\log N))$ by a randomized algorithm.

c) There is a number t and primes $p_1, \ldots, p_t$ such that

1) $t \leq \ln n * N$

2) the mapping $x \to (\ldots((x \bmod p_1) \bmod p_2)\ldots) \bmod p_t)$ operates injectively on S.

3) $p_t \leq O(n^3)$ and

4) $\sum_{i=1}^{t} \ln(p_i) = O(\ln n + \ln \ln N)$ □

For very small load factor, namely $\beta \approx 1/n^2$ and hence $m \approx n^3$, Theorem B answers all three questions. The mapping described in c2 is a perfect hash function for S, it can be evaluated in time $O(\ln * N)$ by c1, it can be found fast by a randomized algorithm (part b) and it has minimal program size (part c4).

Sketch of proof of theorem B: part c) follows from part a) by induction Parts a) and b) follow from the fact that there is a constant c such that at least 50 % of the primes $\leq cn^2 \ln N$ do not divide $\Pi(x_i - x_j)$ and hence satisfy part a). □

Of course, not many people want to work with a load factor as small as $1/n^2$. Load factor $\beta = 1$ is called for. We can only give an almost optimal answer in that case.

Theorem C: Let $\beta = 1$. Then for every $S \subseteq U$, $|S| = n$, there is a program of size $O(n \log n + \log\log N)$ and time complexity $O(\log^* N)$ which computes a perfect hashfunction for S. The program can be found in polynominal time by a randomized algorithm.

Proof: Let h be the function described in theorem B, part c_2. Then $h(S) \subseteq [0\ldots O(n^3)]$ and $|h(S)| = n$. Apply Tarjan and Yao's construction to set h(S). This gives a program of size $O(n \log n)$ and time complexity $O(1)$ computing an injective function from h(S) into $[0\ldots n-1]$. goh is the desired perfect hash function. □

Universal Hashing (Carter/Wegman) ist based on hashing with separate chaining. It deals with the complete dictionary problem. The major and crucial difference with respect to ordinary hashing is that universal hashing works with an entire class H of hashfunctions instead of a single hashfunction. In any particular application the hashfunction to be used is chosen at random from class H. In order for the approach to work class H has to be a c-universal class of hashfunctions for some c.

Definition: Le $c \in \mathbb{R}$. $H \subseteq \{h;\ h : U \to [0..m-1]\}$ is c-universal if for all $x,y \in U$, $x \neq y$

$$|\{h;\ h \in H \text{ and } h(x) = h(y)\}| \leq c \cdot |H|/m$$ □

Note that only a fraction c/m of all functions from a c-universal class leads to collision on any particular pair $x,y \in U$, $x \neq y$. The analysis of Carter/Wegman shows that universal hashing is an $O(c \cdot \beta)$ average case solution to the dictionary problem. The big advantage of universal hashing is the fact that averaging is done over class H and not over the set of possible inputs as in ordinary hashing. The algorithm controls the dice. We consider the following question.

How large must a universal class be?

Carter and Wegman describe a universal class with $\log |H| = O(\log N)$. This was later improved by Wegman/Carter to $\log |H| = O((\log m + \log\log\log N)\log\log N)$. There are several reasons why the size of universal classes is important. First of all, log |H| is a lower bound on the number of coin tosses required to select an element of H. Second of all, log |H| is a lower bound on the program size of a typical member of H. Third for all, Wegman/Carter apply universal hashing in an authentication scheme. In this application U is the set of messages, H is the set of keys and $[0..\ m-1]$ is the set of authentication tags. Two partners in a communication select $h \in H$ at random and exchange it secretly. Then message x U and authentication tag h(x) can be send in the open. Of course, this scheme is only useful if keys are considerably shorter than messages. We show:

Theorem D: a) Let $H \subseteq \{h;\ h : [0\ldots N-1] \to [0\ldots m-1]\}$ be a c-universal class. Then

$$|H| \geq m \cdot (\lceil \log_m N \rceil - 1)/c$$

and hence $\log |H| = \Omega(\log m + \log\log N - \log c)$.

b) Let $N,m \in \mathbb{N}$, $N \geq m$. Then there is a 8-universal class $H \subseteq \{h;\ h : [0...N-1] \to [0...m-1]\}$ of hash functions with

$$\log |H| = O(\log m + \log\log N)$$

Moreover, the elements $h \in H$ can be evaluated in time $O(1)$. □

Sketch of proof: a) Let $H = \{h_1,...,h_t\}$. As in the proof of lemma 1, a) we construct a sequence $U_0 = [0...N-1]$, $U_1, U_2, \ldots$ such that $h_1,\ldots,h_i$ are constant functions on U_i and $|U_i| \geq |U_{i-1}|/m \geq N/m^i$. Let $t_0 = \lceil \log_m N \rceil - 1$. Then $|U_{t_0}| > 1$. Let $x,y \in U_{t_0}$, $x \neq y$. Then

$$t_0 \leq |\{h \in H;\ h(x) = h(y)\}| \leq c \cdot |H|/m$$

since H is c-universal. Thus $|H| \geq m \cdot (\lceil \log_m N \rceil - 1)/c$.

b) Let $N,m \in \mathbb{N}$ and let t be minimal such that $t \cdot \ln p_t \geq m \cdot \ln N$. Here p_t denotes the t-th prime. Then $t = O(m \ln N)$. Let

$$H = \{g_{c,d}(h_\ell(x));\ t < \ell \leq 2t,\ 0 \leq c,d < p_{2t}\}$$

where

$$h_\ell(x) = x \bmod p_\ell$$

and

$$g_{c,d}(z) = [(cz + d) \bmod p_{2t}] \bmod m.$$

Then $|H| = t \cdot p_{2t}^2$ and hence $\log |H_2| = O(\log t) = O(\log m + \log\log N)$ since $\log p_{2t} = O(\log t)$ by the prime number theorem. It remains to show that H is 8-universal. Let $x,y \in U$, $x \neq y$, be arbitrary. We have to count the number of $h \in H$ with $h(x) = h(y)$. Suppose $g_{c,d}(h_\ell(x)) = g_{c,d}(h_\ell(y))$, i. e.

$$[c(x \bmod p_\ell) + d] \bmod p_{2t} = [c(y \bmod p_\ell) + d] \bmod p_{2t} \bmod m$$

Thus there have to exist $q \in [0...m-1]$ and $r,s \in [0...\lceil p_{2t}/m \rceil - 1]$ such that

$$[c(x \bmod p_\ell) + d] \bmod p_{2t} = q + r \cdot m$$

$$[c(y \bmod p_\ell) + d] \bmod p_{2t} = q + s \cdot m$$

We have to count the number of triples (c,d,ℓ) which solve this pair of equations. We count the solutions in two groups. The first group contains all solutions (c,d,ℓ) with $x \bmod p_\ell \neq y \bmod p_\ell$ and the second group contains all solutions (c,d,ℓ) with $x \bmod p_\ell = y \bmod p_\ell$.

Group 1: Of course, there are at most t different ℓ's such that $x \bmod p_\ell \neq y \bmod p_\ell$. For each such ℓ and any choice of q,r and s there is exactly one pair c,d which solves our equations. This follows from the fact that $Z_{p_{2t}}$ is a field. Hence the number of solutions in group one is bounded by

$$t\, m(\lceil p_{2t}/m \rceil)^2 \leq t\, m(1 + p_{2t}/m)^2 \leq (t \cdot p_{2t}^2/m) \cdot (1 + m/p_{2t})^2$$

$$\leq (|H_2|/m) \cdot (1 + m/p_{2t})^2 ;$$

Group 2: Let $L = \{\ell;\ t < \ell \leq 2t \text{ and } x \bmod p_\ell = y \bmod p_\ell\}$ and let $P = \Pi\{p_\ell;\ \ell \in L\}$. Then $P \geq p_t^{|L|}$. Also P divides $x - y$ and hence $P \leq N$. Thus $|L| \leq (\ln N)/\ln p_t \leq t/m$ by definition of t.

Consider any fixed $\ell \in L$ and any choice of q,r and s. If $r \neq s$ then there is no pair (c,d) solving our pair of equations. If $r = s$ then there are exactly p_{2t} pairs (c,d) solving our pair of equations. Hence the number of solutions in group two is at most

$$|L|\, m\lceil (p_{2t}/m) \rceil p_{2t} \leq (t \cdot p_{2t}^2/m)(1 + m/p_{2t}) = (|H_2|/m)(1 + m/p_{2t})$$

Altogether, we have shown that the number of solutions (c,d,ℓ) is bounded by $2(1 + \varepsilon)^2 |H|/m$ where $\varepsilon = m/p_{2t} \leq 1$. (Note that $p_{2t} \leq m$ would imply $t \ln p_t < p_{2t} \ln p_{2t} \leq m \ln m \leq m \ln N$, a contradiction to

the definition of t. Thus H is 8-universal. □

Acknowledgement: This research was started by a intensive discussion with J. Nievergelt. N. Blum provided lemma 1b.

References:

R. Sprugnoli. Perfect Hash Functions: A single probe retrieving method for static sets, CACM 2o, 11 (Nov. 1977), pp. 841-85o.

R.E. Tarjan/A. C-C. Yao. Storing a Sparse Table. CACM, Nov. 1979, Vol 22 , No 11, pp. 6o6-611.

J.L. Carter/M.N. Wegman. Universal Classes of Hash Functions, 9th ACM Symposion on Theory of Computing, 1977, pp. 1o6-112.

M.N. Wegman/J.L. Carter. New Classes and Applications of Hash Functions. 2oto FOCS, 1979, pp. 175-182.

G. Jaeschke. Reciprocal Hashing: A Method for Generating Minimal Perfect Hashing Functions. CACM, Dez. 1981, Vol 24, No 12, pp. 829-833.