Efficient Minimal Perfect Hashing in Nearly Minimal Space.

Torben Hagerup and Torsten Tholey

2001

Abstract

We consider the following problem: Given a subset S of size n of a universe $\{0, ..., u-1\}$, construct a minimal perfect hash function for S, i.e., a bijection h from S to $\{0, ..., n-1\}$. The parameters of interest are the space needed to store h, its evaluation time, and the time required to compute h from S. The number of bits needed for the representation of h, ignoring the other parameters, has been thoroughly studied and is known to be $n\log(e) + \log(\log(u)) \pm O(\log(n))$, where "log" denotes the binary logarithm. A construction by Schmidt and Siegel uses $O(n + \log(\log(u)))$ bits and offers constant evaluation time, but the time to find h is not discussed. We present a simple randomized scheme that uses $n\log(e) + \log(\log(u)) + o(n + \log(\log(u)))$ bits and has constant evaluation time and $O(n + \log(\log(u)))$ expected construction time.

Efficient minimal perfect hashing in nearly minimal space

Torben Hagerup and Torsten Tholey

Institut für Informatik
Johann Wolfgang Goethe-Universität Frankfurt
D-60054 Frankfurt am Main, Germany

Abstract. We consider the following problem: Given a subset S of size n of a universe $\{0, \ldots, u-1\}$, construct a minimal perfect hash function for S, i.e., a bijection h from S to $\{0, \ldots, n-1\}$. The parameters of interest are the space needed to store h, its evaluation time, and the time required to compute h from S. The number of bits needed for the representation of h, ignoring the other parameters, has been thoroughly studied and is known to be $n \log e + \log\log u \pm O(\log n)$, where "log" denotes the binary logarithm. A construction by Schmidt and Siegel uses $O(n + \log\log u)$ bits and offers constant evaluation time, but the time to find h is not discussed. We present a simple randomized scheme that uses $n \log e + \log\log u + o(n + \log\log u)$ bits and has constant evaluation time and $O(n + \log\log u)$ expected construction time.

Key words. Computational and structural complexity, algorithms and data structures, perfect hashing, sparse tables, space complexity.

1 Introduction

Suppose that S is a subset of size n of the universe $\{0, \ldots, u-1\}$ for some $n, u \in \mathbb{N} = \{1, 2, \ldots\}$. A function h defined on $\{0, \ldots, u-1\}$ is said to be *perfect* for S if it is injective on S. If, moreover, the range of h is the set $\{0, \ldots, n-1\}$, then h is called a *minimal perfect hash function* for S. We consider the problem of constructing minimal perfect hash functions for given sets of nonnegative integers.

Let $\mathcal{A}$ be an algorithm that inputs an arbitrary set S of nonnegative integers and outputs a minimal perfect hash function h for S. Several performance parameters of $\mathcal{A}$ are of interest:

- *Encoding size:* The number of bits of storage occupied by the representation of h output by $\mathcal{A}$.
- *Evaluation time:* The time needed to compute $h(x)$ for an arbitrary x in the domain of h.
- *Construction time:* The time needed to compute h from S.
- *Working space:* The amount of space needed to compute h from S.

We view these parameters as functions of $n = |S|$ and $u = 1 + \max S$. Fredman, Komlós and Szemerédi described a randomized construction that achieves $O(n \log u)$ encoding size, $O(1)$ evaluation size and $O(n)$ expected construction time [4]. Strictly speaking, their scheme yields a function h whose range is of size $O(n)$ rather than n, but is it easy to obtain a minimal perfect hash function within the same resource bounds. Using a counting argument, Fredman and Komlós proved a worst-case lower bound of $n \log e + \log\log u - O(\log n)$ bits for the encoding size of a minimal perfect hash function for a subset of size n of a universe of size u, provided that $u \geq n^{2+\epsilon}$ for some fixed $\epsilon > 0$ [3] (an easy alternative proof was given by Radhakrishnan [10]). That this bound is almost tight follows by comparing it with an upper bound of $n \log e + \log\log u + O(\log n)$ bits given by Mehlhorn [8, Sect. III.2.3, Thm. 8]. His construction, however, has $n^{\Theta(ne^n u \log u)}$ construction and evaluation time. Schmidt and Siegel showed the existence of minimal perfect hash functions combining an encoding size of $O(n + \log\log u)$ bits with $O(1)$ evaluation time, but the time needed to find such functions was not discussed [11]. We present a new construction that not only works in almost linear expected time while still offering constant-time evaluation, but also reduces the encoding size to within lower-order terms of the lower bound.

Our model of computation is a unit-cost word RAM [5] with an instruction set including multiplication and integer division. We denote the word length of the machine by w and assume that every input set S consists of numbers representable in single words, i.e., $\max S < 2^w$. We will measure the encoding size of a hash function in bits, but the working space needed for its construction in w-bit words. Our main result is expressed in the following theorem.

Theorem 1. *For all integers $n, u, w \geq 4$ with $u \leq 2^w$ and for every given subset S of size n of $\{0, \ldots, u-1\}$, a minimal perfect hash function for S that can be evaluated in $O(1)$ time and stored in $n \log e + \log\log u + O(n(\log\log n)^2/\log n + \log\log\log u)$ bits can be constructed in $O(n + \log\log u)$ expected time using $O(n)$ words of working space on a unit-cost word RAM with a word length of w bits and an instruction set including multiplication and integer division.*

Our approach is very simple. Suppose that we are given an input set S of size n. Repeatedly replacing the elements of S by their remainders modulo suitable primes, we begin by mapping S bijectively to a set S' whose elements are either bounded by a polynomial in n or far smaller than $\max S$. In the former and more interesting case, we proceed to partition S' into groups of elements small enough to be handled by the doubly exponential algorithm of Mehlhorn mentioned above. The division into groups is done in two stages, each of which defines a group as the set of elements mapped to a common value by a suitable hash function, a so-called *bucket* of the hash function. The hash functions employed for this purpose have to be chosen rather carefully, as the maximum bucket size must be within a constant factor of the average bucket size. Essential in achieving a construction time that is linear and not merely almost linear in n is the observation that the superlinear component in the running time of Mehlhorn's algorithm can be amortized over all groups.

2 Reducing the Size of the Universe

We denote by the term *range reduction* the process of reducing an instance of the problem at hand, namely computing a minimal perfect hash function for a given set S, to an instance that involves smaller input numbers, i.e., the process of reducing the size of the universe. We employ a range reduction based on the following lemma, proved essentially as [4, Lemma 2].

Lemma 2. *There is a constant $\beta \in \mathbb{N}$ such that for every nonempty set S of nonnegative integers and for every $m \geq \beta |S|^2 \log(1 + \max S)$, the function $x \mapsto x \bmod p$ is injective on S for at least half of the primes p bounded by m.*

Let S be an input set and take $n = |S|$, $u = 1 + \max S$, $\lambda = \beta n^2 \lceil \log u \rceil$ and $D = \{p \in \mathbb{N} \mid p \leq \lambda \text{ and the function } x \mapsto x \bmod p \text{ is injective on } S\}$. We assume that $n \geq 4$. In order to put Lemma 2 to use, we need a way to compute an element of D.

If $\log u$ and therefore λ are polynomial in n, we can pick an integer p uniformly at random from $M = \{1, \ldots, \lambda\}$, apply Rabin's randomized primality test [9] to p $\lceil \log n \rceil$ times and, if p passes this test—which happens with probability at most $1/n$ if p is composite—proceed to test directly whether $p \in D$ by means of radix sorting. If p fails any test, we immediately discard it and pick a new random integer, continuing until an element of D is encountered. By Lemma 2, the expected number of trials in which p is prime is bounded by a constant, and the expected time spent in such trials is $O(n)$. By the prime number theorem, the density of primes in M is $\Omega(1/\log \lambda) = \Omega(1/\log n)$, so that the expected total number of trials is $O(\log n)$. Since Rabin's test works in $(\log n)^{O(1)}$ time, the total expected time is $O(n)$.

For $\log u \geq n^3$, we sketch a different method and allow an expected time of $O(n + \log \lambda) = O(n + \log \log u)$. Note first that λ can be computed within this time bound. We pick a set R of $\lceil \log \lambda \rceil$ random elements of M and store these, each replicated $\lfloor \sqrt{\lambda} \rfloor$ times, together in a single computer word A. The condition $\log u \geq n^3$ ensures that the word length is sufficient for this to be possible (unless u is smaller than some constant). We also create a word B containing the sequence $1, \ldots, \lfloor \sqrt{\lambda} \rfloor$, replicated $\lceil \log \lambda \rceil$ times, and proceed to divide each number in A by the corresponding number in B. Simulating the school method for long division, this can be carried out simultaneously for all pairs of numbers in $O(\log \lambda)$ time; a more detailed discussion of similar computations can be found in [5]. As a result, we learn for each element of R whether it has a divisor bounded by $\lfloor \sqrt{\lambda} \rfloor$, i.e., whether it is composite. The set R was chosen sufficiently large to ensure that with probability $\Omega(1)$ it contains at least one prime. If this is the case, we pick such a prime p and test whether $p \in D$. Because p is much smaller than u, this can be done in $O(n)$ time by sorting [5]; alternatively, it can be done in $O(n)$ expected time using universal hashing [1]. If no element of D is found, we repeat the entire procedure. Since each trial takes $O(n + \log \log u)$ time and succeeds with probability $\Omega(1)$, the overall expected time is $O(n + \log \log u)$.

Faced with an input set S with $|S| = n$, we repeatedly apply the reduction based on Lemma 2 and discussed above until we reach a set S' with $\max S' \leq n^3$,

but at most four times. The expected time to do this is $O(n + \log\log u)$, and $O(n)$ words of working space suffice. The first reduction requires a prime of at most $\log\lambda = \log\log u + O(\log n)$ bits to be stored as part of the representation of the final minimal perfect hash function. The number of bits required for all following reductions is $O(\log n + \log\log\log u)$.

After four reduction steps, we have a set S' with $\max S' = O(n^2(\log^{(3)} n + \log^{(4)} u))$. If the condition $\max S' \leq n^3$ is still not satisfied, $n = O(\log^{(4)} u)$, and we can simply store S' using the method of Fredman, Komlós and Szemerédi [4], which requires $O(n \log \max S') = O((\log^{(4)} u)^4)$ bits of storage, for a total of $\log\log u + o(\log\log\log u)$ bits. In the following, we can therefore assume without loss of generality that the input set S satisfies $\max S \leq n^3$.

3 Splitting into Groups

Our goal in this section is to partition the set S into $O(n/\hat{n})$ groups of at most $\hat{n}$ elements each, where $\hat{n} = \gamma \log n / \log\log n$ for a constant $\gamma > 0$ to be chosen later. Our main tool is a class $\mathcal{R}$ of hash functions introduced by Dietzfelbinger and Meyer auf der Heide [2] (another possibility would be to use a class defined by Siegel [12]). For our purposes, the distinguishing feature of $\mathcal{R}$ is that a function drawn at random from $\mathcal{R}$ is likely to spread a key set about evenly over its range.

We begin by defining the class $\mathcal{R}$. Fix a prime $p \geq u$, let $U = \{0, \ldots, p-1\}$ and, for $d, s \in \mathbb{N}$, take

$$\mathcal{H}_s^d = \{h_a \mid a = (a_0, \ldots, a_d) \in U^{d+1}\},$$

where, for $a = (a_0, \ldots, a_d) \in U^{d+1}$, $h_a : U \to \{0, \ldots, s-1\}$ is the function given by

$$h_a(x) = \left(\sum_{i=0}^{d} a_i x^i \bmod p \right) \bmod s$$

for all $x \in U$. Informally, $\mathcal{H}_s^d$ is known as the class of polynomials of degree d. The class $\mathcal{R}$ depends on four parameters $r, s, d_1, d_2 \in \mathbb{N}$, a dependence made explicit by writing $\mathcal{R}$ as $\mathcal{R}(r, s, d_1, d_2)$. For $r, s, d_1, d_2 \in \mathbb{N}$,

$$\mathcal{R}(r, s, d_1, d_2) = \{h_{(f,g,a_0,\ldots,a_{r-1})} \mid f \in \mathcal{H}_r^{d_1}, g \in \mathcal{H}_s^{d_2} \text{ and } 0 \leq a_0, \ldots, a_{r-1} < s\},$$

where, for $f \in \mathcal{H}_r^{d_1}$, $g \in \mathcal{H}_s^{d_2}$ and $a_0, \ldots, a_{r-1} \in \{0, \ldots, s-1\}$, $h_{(f,g,a_0,\ldots,a_{r-1})} : U \to \{0, \ldots, s-1\}$ is the function given by

$$h_{(f,g,a_0,\ldots,a_{r-1})}(x) = (g(x) + a_{f(x)}) \bmod s,$$

for all $x \in U$. One way to visualize $\mathcal{R}$ is as follows: A key $x \in U$ is first mapped to row $f(x)$ and column $g(x)$ of an $r \times s$ table. Then row i is rotated cyclically a distance of a_i, for $i = 0, \ldots, r-1$, and the resulting column number is taken as the final function value.

The nontrivial fact about $\mathcal{R}$ of interest to us is expressed in the following lemma, related to Lemma 4.4 and Theorem 4.6 of [2].

Lemma 3. *For every $c > 0$, there is a $C > 0$ such that for all $r, s, d_1, d_2 \in \mathbb{N}$ with $r \leq n$, $s \leq cn/\log n$, $rs \geq n^{1+1/c}$, $d_1 \geq C$ and $d_2 \geq C$, the relation*

$$\forall i \in \{0, \ldots, s-1\} : \quad |\{x \in S \mid h(x) = i\}| \leq Cn/s$$

holds with probability at least $1 - n^{-1}$ if h is chosen uniformly at random from $\mathcal{R}(r, s, d_1, d_2)$.

Informally, the lemma says that if r and s are chosen so that $rs = \Omega(n^{1+\epsilon})$ for some fixed $\epsilon > 0$ and $s = O(n/\log n)$, then for sufficiently large d_1 and d_2, the maximum bucket size of a random function from $\mathcal{R}(r, s, d_1, d_2)$ will be within a constant factor of the average bucket size, except with negligible probability.

We prove Lemma 3 using several auxiliary lemmas. Note that we can assume n to be larger an arbitrary constant, since the maximum bucket size is trivially bounded by Cn/s if $C \geq kn$. In particular, we assume that $s \leq n$.

Let $\xi = n^{1+1/(2c)}/r$. We begin by showing, using the following lemma, that if $f \in \mathcal{H}_r^{d_1}$ is chosen uniformly at random and d_1 is sufficiently large, then the maximum bucket size $\max_{0 \leq j < r} |\{x \in S \mid f(x) = j\}|$ of f is bounded by 2ξ, except with negligible probability.

Lemma 4. *Let $n, d \in \mathbb{N}$, let $X_1, \ldots, X_n$ be d-independent, equidistributed 0-1-variables, let $\mu \geq E(X_1)$ and assume that $n\mu \geq d$. Then, for some α that depends only on d and for every $\xi > 0$,*

$$\Pr\left(\sum_{i=1}^{n}(X_i - \mu) > \xi\right) \leq \frac{\alpha(n\mu)^{d/2}}{\xi^d}.$$

The lemma is essentially [7, Corollary 4.20]. We generalize the original formulation in a trivial way by allowing $\mu \geq E(X_1)$ instead of taking $\mu = E(X_1)$ and replace the original condition $n \geq d/(2\mu)$ by the stronger condition $n\mu \geq d$, which seems called for by the proof.

In our context, with $S = \{x_1, \ldots, x_n\}$, we fix $j \in \{0, \ldots, r-1\}$ and take

$$X_i = \begin{cases} 1, & \text{if } f(x_i) = j \\ 0, & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, n$. Then $X_1, \ldots, X_n$ satisfy the conditions of Lemma 4 with $d = d_1$ and $\mu = 2/r + d_1/n$. For every d_1 and for sufficiently large n, we have $\xi \geq n\mu$, and therefore the quantity $|\{x \in S \mid f(x) = j\}| = \sum_{i=1}^{n} X_i$ is bounded by 2ξ, except with probability at most $\alpha\xi^{-d_1/2}$, where α depends only on d_1. For d_1 and subsequently n chosen sufficiently large, the latter probability is at most n^{-3}, so $\max_{0 \leq j < r} |\{x \in S \mid f(x) = j\}| > 2\xi$ with probability at most $rn^{-3} \leq n^{-2}$.

Assuming that f has been chosen so that its maximum bucket size is indeed bounded by 2ξ, we next show that if $g \in \mathcal{H}_s^{d_2}$ is chosen uniformly at random and d_2 is sufficiently large, then for each application of g to a bucket of f, the maximum bucket size is bounded by d_2, except with negligible probability.

Lemma 5 ([2, Fact 2.2(b)]). *For all $m, s, d \in \mathbb{N}$ and for every subset B of U of size m, if g is chosen uniformly at random from $\mathcal{H}_s^d$, then $\max_{0 \le i < s} |\{x \in B \mid g(x) = i\}| \le d$ with probability at least $1 - m \cdot (2m/s)^d$.*

We use the lemma for $j = 0, \ldots, r-1$ with $B = B_j = \{x \in S \mid f(x) = j\}$. In our case, $d = d_2$, and $m \le 2\xi$, so that $2m/s \le 4n^{1+1/(2c)}/(rs) \le 4n^{-1/(2c)}$. Thus, if d_2 and subsequently n are chosen sufficiently large, then $\Pr(|\{x \in B \mid g(x) = i\}| \ge d_2) \le n^{-3}$ for $i = 0, \ldots, s-1$ and $\Pr(\max_{0 \le i < s} |\{x \in B \mid g(x) = i\}| \ge d_2) \le sn^{-3} \le n^{-2}$.

We now come to the proof of Lemma 3 itself. Concerning the random choice of h, we assume that f and g have already been selected, so that only $a_0, \ldots, a_{r-1}$ remain to be chosen. Then, for each fixed $i \in \{0, \ldots, s-1\}$, the quantity $Z_i = |\{x \in S \mid h(x) = i\}|$ is the sum of independent random variables $X_0, \ldots, X_{r-1}$ where, for $j = 0, \ldots, r-1$, $X_j = |\{x \in B_j \mid h(x) = i\}|$. It is easy to see that $E(Z_i) = n/s$. We will assume that $X_j \le d_2$ for $j = 0, \ldots, r-1$; by what was shown above, this ignores an event of negligible probability.

Lemma 6 (Hoeffding; see [6, p. 104]). *Let Z be a sum of independent non-negative random variables, each bounded by $z > 0$, and take $\mu = E(Z)$. Then, for all $t > 0$,*

$$\Pr(Z \ge \mu + t) \le \left[\left(\frac{\mu}{\mu + t} \right)^{\mu + t} e^t \right]^{1/z}.$$

Using the lemma with $z = d_2$, $\mu = n/s$ and $t = (C-1)\mu$, we obtain

$$\Pr(Z_i \ge Cn/s) \le (e/C)^{Cn/(d_2 s)} \le (e/C)^{(C \log n)/(d_2 c)}.$$

For sufficiently large C, we have $\Pr(Z_i \ge Cn/s) \le n^{-3}$ and $\Pr(\max_{0 \le i < s} Z_i \ge Cn/s) \le sn^{-3} \le n^{-2}$. Adding the three "error" probabilities identified above, we see that the assertion of Lemma 3 holds with probability at least $1 - 3n^{-2} \ge 1 - n^{-1}$. This ends the proof of Lemma 3.

The condition $s = O(n/\log n)$ of Lemma 3 prevents us from splitting S into groups of size at most $\hat{n}$ in one go. We therefore begin by splitting S into groups of size $O((\log n)^3)$. We take $r = \Theta(\sqrt{n})$, $s = \Theta(n/(\log n)^3)$ and C, d_1 and d_2 according to Lemma 3 (for $c = 3$, say) and repeatedly choose $h \in \mathcal{R}(r, s, d_1, d_2)$ uniformly at random until $\max_{0 \le i < s} |\{x \in S \mid h(x) = i\}| \le Cn/s$. By Lemma 3, the expected number of trials is $O(1)$, and the computation can be carried out in $O(n)$ expected time using $O(n)$ words of working space. By the assumption $\max S \le n^3$, the chosen function h can be represented in $O(r \log n) = O(\sqrt{n} \log n)$ space, and its evaluation takes $O(1)$ time.

For each of the resulting groups of size at most $l = \Theta((\log n)^3)$, we use a single range reduction based on Lemma 2 to force all integers in the group below an integer $\hat{v}$ with $\hat{v} = (\log n)^{O(1)}$. This requires the storage of one prime of $O(\log \log n)$ bits per group, for a total of $o(n/\log n)$ bits.

Our remaining task is to reduce the group size further from at most l to at most $\hat{n} = \gamma \log n / \log \log n$. We do this by another application of Lemma 3

with $r = \Theta((\log n)^2)$ and with $s \geq Cl/\hat{n}$ (ensuring that each group is split into subgroups of at most $\hat{n}$ elements each), but $s = O(l/\hat{n})$ (ensuring that the total number of subgroups is $O(n/\hat{n})$). The space needed for storing the required functions in $\mathcal{R}$ is $O(r \log\log n)$ bits per group, for a total of $O(n \log\log n/\log n)$ bits.

4 Perfect Hashing by Brute Force

In the previous section we reduced the original problem of size n to a collection of subproblems of size $O(\log n/\log\log n)$ and involving numbers of size polylogarithmic in n. In this section we discuss how to solve a single such subproblem using nearly minimal space.

Fix integers $m, v \geq 2$ and let $\mathcal{F}$ be the set of all functions from $V = \{0, \ldots, v-1\}$ to $\{0, \ldots, m-1\}$. We call a (multi)subset F of $\mathcal{F}$ *perfect* if for every subset B of V of size m, F contains a (minimal) perfect hash function for B. For $t \in \mathbb{N}$, denote by $q(t)$ the probability that a multiset of t random functions drawn independently from the uniform distribution over $\mathcal{F}$ is not perfect. In the proof of [8, Sect. III.2.3, Thm. 7], Mehlhorn argues that

$$q(t) \leq \binom{v}{m}\left(1 - \frac{m!}{m^m}\right)^t$$

for all $t \in \mathbb{N}$ and proves that the right-hand side is smaller than 1 for $t = t^* = \lceil me^m \ln v \rceil$.

It follows that there exists a perfect set of size t^*, and Mehlhorn proceeds to define a canonical such set F^* as the first perfect set encountered in some fixed enumeration of the subsets of $\mathcal{F}$ of size t^*. Because F^* can be recalculated for every query, it need not be stored, which is crucial in the original setting. In our application, however, m and v are tiny, and storing a perfect set is feasible. This allows us to replace the deterministic procedure of [8], which runs in doubly-exponential time, by a randomized procedure whose running time is merely singly exponential.

We first observe that since $m! \geq (m/e)^m$,

$$q(t^*+1) \leq q(t^*)\left(1 - \frac{m!}{m^m}\right) \leq 1 - e^{-m}.$$

It follows that if we repeatedly draw a multiset of $t^* + 1$ random functions from $\mathcal{F}$ until a perfect multiset is encountered, then the expected number of trials is $O(e^m)$. Moreover, each multiset can be tested for perfectness in time $O\left(\binom{v}{m} m(t^*+1)\right) = O(v^m e^m \ln v)$, so a perfect multiset F^* of $t^* + 1$ functions from $\mathcal{F}$ can be found in $v^{O(m)}$ expected time. As a by-product of the computation, we discover for each subset B of V of size m a function in F^* that is perfect for B.

Lemma 7. *Given integers $m, v \geq 2$ and a subset B of size m of $\{0, \ldots, v-1\}$, $v^{O(m)}$ expected time and $v^{O(m)}$ words of working space suffice to compute a*

minimal perfect hash function for B that can be evaluated in constant time and whose representation consists of $v^{O(m)}$ bits that depend only on m and v and $m \log e + \log\log v + O(\log m)$ bits that depend also on B.

Proof. Carry out the construction described above and store the set F^*, which depends only on m and v, as well as a pointer of $\log |F^*| = m \log e + \log\log v + O(\log m)$ bits to a function in F^* that is perfect for B.

5 The Complete Construction

In Section 3 the problem of hashing the original set S was reduced to that of hashing groups $G_1, \ldots, G_k$, where $|G_i| \leq \hat{n}$ and $\max G_i < \hat{v}$ for $i = 1, \ldots, k$ and $k = \Theta(n/\hat{n})$. More precisely, we showed that, within the resource bounds of Theorem 1, we can map S injectively to a set $\hat{S}$ of pairs $(i, j) \in \{1, \ldots, k\} \times \{0, \ldots, \hat{v} - 1\}$ such that for $i = 1, \ldots, k$, $|G_i| \leq \hat{n}$, where $G_i = \hat{S} \cap (\{i\} \times \{0, \ldots, \hat{v} - 1\})$. Moreover, for a certain constant $\rho \geq 1$, we showed in Section 4 how to compute a minimal perfect hash function h_i for G_i in at most $\hat{v}^{\rho\hat{n}}$ steps, for $i = 1, \ldots, k$, such that the representation space of h_i consists of at most $\hat{v}^{\rho\hat{n}}$ bits that depend only on $|G_i|$ (the *shared part*) and $|G_i| \log e + O(\log\log n)$ bits that depend also on G_i (the *individual part*). We still need to describe how to combine the solution for single groups to a solution for the full set S.

Fix a constant ν so that $\hat{v} \leq (\log n)^\nu$ and recall that $\hat{n} = \gamma \log n / \log\log n$ for a constant $\gamma > 0$ that can still be chosen freely. Now

$$\hat{v}^{\rho\hat{n}} \leq 2^{\nu \log\log n \cdot \rho \cdot \gamma \log n / \log\log n} = n^{\nu\rho\gamma}.$$

We choose $\gamma = 1/(3\nu\rho)$, which makes $\hat{v}^{\rho\hat{n}} \leq n^{1/3}$. Then, since the number of possible distinct groups is bounded by $1 + \hat{v}^{\hat{n}} = O(n^{1/3})$, we can compute minimal perfect hash functions for all possible groups in $O(n^{2/3})$ time. We compute a table mapping each possible group size to the corresponding public part and another table mapping each group, represented as an integer of size $O(n^{1/3})$, to the corresponding individual part. The space needed for these tables is negligible. We create another table L mapping each $i \in \{1, \ldots, k\}$ to $|G_i|$, which allows the public part of h_i to be accessed, and to the individual part of h_i. The entries in L can be computed in $O(n)$ time, and their total size is

$$\sum_{i=1}^{k} (|G_i| \log e + O(\log\log n)) = n \log e + O(n(\log\log n)^2 / \log n)$$

bits, in accordance with Theorem 1. We still have to solve two problems, however:

(1) We cannot use the functions $h_1, \ldots, h_k$ directly, because their ranges overlap. Rather, we would like to replace h_i by $h_i + \sum_{j=1}^{i-1} |G_j|$, for $i = 1, \ldots, k$.
(2) Because the entries in L are not all of the same length, it is not clear how to access the ith entry in constant time.

The following lemma provides a solution to these problems.

Lemma 8. *For all integers m, $N \geq 4$, given m integers $a_1, \ldots, a_m$ with $\sum_{i=1}^{m} a_i \leq N$, a data structure that occupies $O(m(\log\log m + \log(1 + N/m)))$ bits and allows the computation of $b_i = \sum_{j=1}^{i} a_j$ from i in constant time, for $i = 1, \ldots, m$, can be constructed in $O(m)$ time and space.*

Proof. If $N > m^2$, we can simply store $b_1, \ldots, b_m$ in a table with m entries of $\lceil \log(N+1) \rceil$ bits each. Assume therefore that $N \leq m^2$. Our data structure is a tree T of depth 2 with at least m leaves in which every node of depth 1 has $d = O(\log m)$ children and the root has $O(m/\log m)$ children. Conceptually, we label the ith leaf of T, counted from the left, with a_i, for $i = 1, \ldots, m$, and the remaining leaves with zero. For every node v of T, denote by $s(v)$ the sum of the labels at leaves that are descendants of left siblings of v or equal to v. For $i = 1, \ldots, m$, the prefix sum b_i is the sum of $s(v)$ over all ancestors v of the ith leaf of T.

We call a leaf v *good* if $s(v) \leq (N/m)(\log m)^2$, and *bad* otherwise, If a leaf v is good, we store $s(v)$ in a field of $O(\log\log m + \log(1 + N/m))$ bits associated with v. Similarly, for each internal node v, we store $s(v)$ in a field of $O(\log m)$ bits associated with v. Together, these fields occupy $O(m(\log\log m + \log(1 + N/m)))$ bits.

Call a node v of depth 1 good if all of its children are good, and bad otherwise. For each bad node v of depth 1, we store all the values $s(y)$, where y is a child of v, in a table with d fields of $O(\log m)$ bits each in an overflow area and store a pointer to this table with v. Since the number of bad nodes of depth 1 is bounded by $m/(\log m)^2$, an overflow area of size $O(m)$ suffices. Altogether, the space needed is $O(m(\log\log m + \log(1 + N/m)))$ bits, and it is easy to see that b_i can be computed in constant time from i for $i = 1, \ldots, m$.

In order to solve Problem (1), we store the groups sizes $|G_1|, \ldots, |G_k|$ using the method of Lemma 8. Since $\sum_{i=1}^{k} |G_i| = n$, the space needed comes to $O(k(\log\log n + \log(1 + n/k))) = O(n(\log\log n)^2/\log n)$. In order to solve Problem (2), we store the individual parts of $h_1, \ldots, h_k$ as one contiguous bit string W and store the sizes of these individual parts using the method of Lemma 8, which allows us to pick out any individual part from W in constant time. Since the total size of all individual parts is $O(n)$, the space needed again is $O(n(\log\log n)^2/\log n)$. This ends the proof of Theorem 1.

References

1. J. L. Carter and M. N. Wegman, Universal Classes of Hash Functions, *J. Comput. System Sci.* **18** (1979), pp. 143–154.
2. M. Dietzfelbinger and F. Meyer auf der Heide, A new universal class of hash functions and dynamic hashing in real time, Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP 1990), Lecture Notes in Computer Science, Vol. 443, Springer-Verlag, Berlin, pp. 6–19.
3. M. L. Fredman and J. Komlós, On the size of separating systems and families of perfect hash functions, *SIAM J. Alg. Disc. Meth.* **5** (1984), pp. 61–68.

4. M. L. Fredman, J. Komlós and E. Szemerédi, Storing a sparse table with $O(1)$ worst case access time, *J. ACM* **31** (1984), pp. 538–544.
5. T. Hagerup, Sorting and searching on the word RAM, Proc. 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1998), Lecture Notes in Computer Science, Vol. 1373, Springer-Verlag, Berlin, pp. 366–398.
6. M. Hofri, *Probabilistic Analysis of Algorithms*, Springer-Verlag, New York, 1987.
7. C. P. Kruskal, L. Rudolph and M. Snir, A complexity theory of efficient parallel algorithms, *Theoret. Comput. Sci.* **71**, (1990), pp. 95–132.
8. K. Mehlhorn, *Data Structures and Algorithms, Vol. 1: Sorting and Searching*, Springer-Verlag, Berlin, 1984.
9. M. O. Rabin, Probabilistic algorithm for testing primality. *J. Number Theory* **12**, (1980), pp. 128–138.
10. J. Radhakrishnan, Improved bounds for covering complete uniform hypergraphs, *Inform. Process. Lett.* **41** (1992), pp. 203–207.
11. J. P. Schmidt and A. Siegel, The spatial complexity of oblivious k-probe hash functions, *SIAM J. Comput.* **19** (1990), pp. 775–786.
12. A. Siegel, On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications, Proc. 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1989), pp. 20–25.