Average case analysis for a simple compression algorithm

Donatella Merlini, Renzo Sprugnoli, M. Cecilia Verri
Dipartimento di Sistemi e Informatica
via Lombroso 6/17, 50134
Firenze, Italy

Abstract

In this paper, we treat the *static dictionary problem*, very well-known in computer science. It consists in storing a set S of m elements in the range $[1..n]$ so that membership queries on S's elements can be handled in $O(1)$ time. It can be approached as a *table compression problem* in which a size n table has m 1's and the other elements are 0's. We focus our attention on sparse cases ($m \ll n$). We use a simple algorithm to solve the problem and make an average case analysis of the total space required when the input derives from uniform probability distribution. We also find some conditions able to minimize storage requirements. We then propose and analyze a new algorithm able to drastically reduce storage requirements to $O(m^{4/3})$.

1 Introduction

The *static dictionary problem* (very well-known in computer science) consists in storing a set S of m elements in the range $[1..n]$ in such a way that we are able to retrieve an item belonging to S in constant time. In this paper, we examine the sparse cases: $m \ll n$. Among other things, this problem has been studied with regard to the storage of transition functions of finite state automaton [1] and sparse Gaussian elimination [7].

The simplest solution corresponds to $O(1)$-time membership queries and is obtained by storing set S in a size n array V whose ith entry is marked 1 if $i \in S$. However, this solution is wasteful when $m \ll n$. When sparse cases are involved, we can approach the issue as a *table compression problem*: we look for a function $f : [1..n] \to [1..n']$, in which $n' = O(m)$ is the size of the table used and f is computable in $O(1)$ time. We can solve the problem by perfect hashing (see for example [**?**, 6]) or various other techniques, such as in [4].

In Section 2, we use a simple algorithm to solve this problem: we proceed by fixing a parameter s and dividing array V into some non-overlapping size s *segments*. We then cut off the leading and trailing 0's of each segment and join the remaining *truncated segments* into a *compressed array* C. We then store the following two quantities into auxiliary arrays F and Z: each segment's starting location in C and the number of leading 0's.

This takes us $2(n/s)$ storage words. If we want to retrieve an element, we have to perform an integer division by s in order to determine the segment to be searched. We also perform two memory indirections (at most) in order to locate C's query element. This algorithm has been recently used in [2].

The storage space required for array C is a fundamental element in our study. In Section 3, we examine the algorithm when its input derives from uniform probability distribution and make an average case analysis of the total space the algorithm uses. We do this by showing that each segment can be defined by a regular expression to which we can apply the Schützenberger methodology (see [3, 5]). We obtain a multivariate generating function $S(t, u, w)$ that describes the set of all the 0/1 sequences that form a segment. This function enables us to determine the average space required by the method, both as an exact formula (Theorem 3.1) and as an asymptotic value when the ratio $c = m/n$ remains constant but $m, n \to \infty$ (Section 4).

Unfortunately, our analysis shows that the space occupancy is $O(n^{1/3}m^{2/3})$, which is too large for practical purposes. In Section 5, we therefore propose a new algorithm to reduce storage requirements and, by choosing a suitable parameter p, we are able to limit them to $O(m^{4/3})$ without changing the constant time performance of the searching strategy. We wish to add that we found a computer algebra system like Maple a virtually essential tool in our work.

2 The algorithm

We consider a set S of m elements in the range $[1..n]$ and assume they occupy the locations of an array $V[1..n]$ (we can image that V's ith entry is marked 1 if $i \in S$, 0 otherwise). The algorithm builds a compressed array C, which includes V's non-zero elements, and two auxiliary arrays $F[1..r]$, $Z[1..r]$, which allow us to retrieve any V element in constant time.

We divide the array V into $r = \lceil n/s \rceil$ s-length segments. Without any loss in generality, we can assume that s divides n; if this is not so, the last segment can be padded with trailing 0's and n can be incremented to $s\lceil n/s \rceil$. For simplicity's sake, we always assume $n = rs$, for some $r \in \mathbf{N}$.

Any segment of V contains some leading and trailing zeroes; we build C by deleting the zeroes and concatenating the truncated segments. We define arrays F and Z in the following way: for $1 \le i \le r$, $F[i]$ is the address of the ith truncated segment's first element in array C (if the truncated segment is empty, we use the place such an element would have appeared as its address) and $Z[i]$ is the number of leading zeroes in the ith untruncated segment. We wish to point out that the ith truncated segment's length is $F[i+1]-F[i]$. The construction of C, F and Z is at most $O(n)$ if we perform a scanning for V. Figure 2.1 illustrates an example of this construction for $n = 99$, $m = 12$, $s = 9$ and $S = \{8, 10, 21, 36, 38, 39, 43, 51, 56, 58, 68, 73\}$: vector V, whose elements are grey, is subdivided into 11 segments; on each segment's right we also indicate the corresponding truncated segment. Given $j \in [1..n]$, we want to retrieve $V[j]$ from C by using the information stored in F and Z. We can see that $V[j]$ belongs to the hth untruncated segment of V, starting from the left, where $h = \lceil j/s \rceil$. If we let d be the address of $V[j]$ relative to the beginning of the hth untruncated segment (starting from 0), we get $d = (j-1) \bmod s$. We have the following three cases:

1. $0 \le d < Z[h]$: in this case, d corresponds to a leading zero and $V[j] = 0$;

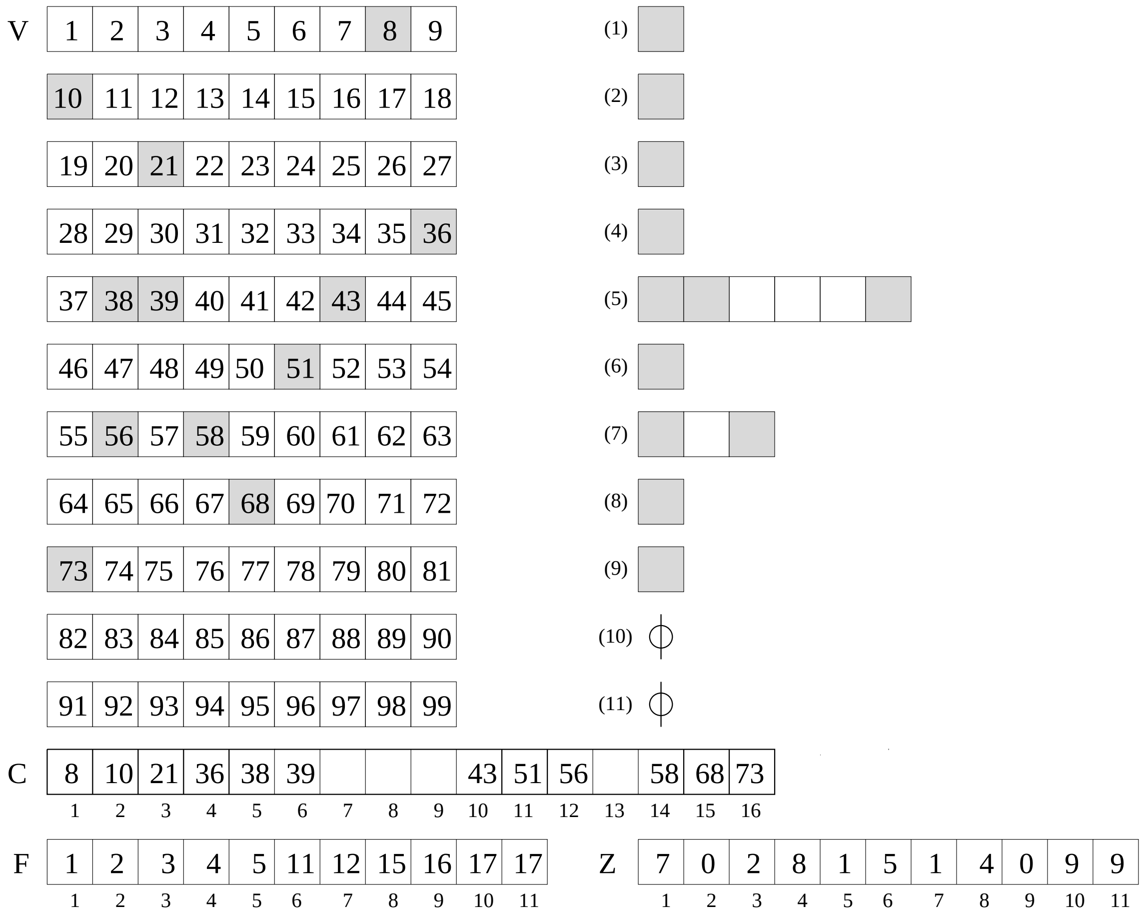

Figure 2.1: The construction of C, F and Z.

2. $Z[h] \leq d < Z[h] + F[h+1] - F[h]$; in this case, d corresponds to an element of the truncated segment, in particular $V[j] = C[F[h] + d - Z[h]]$;

3. $Z[h] + F[h+1] - F[h] \leq d < s$: in this case, d corresponds to a trailing zero and $V[j] = 0$.

If we want to retrieve the value 36 shown in Figure 2.1, we find $h = 4$, $d = 8$, and therefore $V[36] = C[4]$, that is, number 36 belongs to the set S.

3 Average case analysis

In Figure 3.1, we illustrate all the possible V arrays arising in a very small problem: $n = 6$, $m = 2$ and $s = 3$ (and therefore $r = 2$). Obviously, there are $\binom{6}{2} = 15$ possibilities and we explicitly make the hypothesis that our analysis concerns the uniform case, i.e., in the example, each of the 15 combinations has the same probability ($p = 1/15$) of occurring. The numbers in parentheses in Figure 3.1 represent array C's size for each V : C's average size in this case is $32/15 = 2.133$.

It is worth noting that the regular expression $\{0,1\}^*$ (representing all the possible 0-1-sequences) can be decomposed into $0^*1^+\{0^+1^+\}^*0^* + 0^*$ to separate the leading and trailing

(2) 1 1 0 | 0 0 0	(3) 1 0 1 | 0 0 0	(2) 1 0 0 | 1 0 0
(2) 1 0 0 | 0 1 0	(2) 1 0 0 | 0 0 1	(2) 0 1 1 | 0 0 0
(2) 0 1 0 | 1 0 0	(2) 0 1 0 | 0 1 0	(2) 0 1 0 | 0 0 1
(2) 0 0 1 | 1 0 0	(2) 0 0 1 | 0 1 0	(2) 0 0 1 | 0 0 1
(2) 0 0 0 | 1 1 0	(3) 0 0 0 | 1 0 1	(2) 0 0 0 | 0 1 1

Figure 3.1: The 15 possible V arrays for $n = 6, m = 2$ and $s = 3$.

zeroes. We then mark any zero and one by t, any one by u, and any zero and one by w when they belong to the truncated segment. By using standard methods (e.g., Schützenberger's methodology [5]) to go from regular expressions to counting generating functions, we immediately get:

$$S(t,u,w) = \frac{1}{(1-t)^2}\frac{tuw}{1-tuw}\frac{1}{1-\frac{tw}{1-tw}\frac{tuw}{1-tuw}} + \frac{1}{1-t},$$

where the coefficient $[t^s u^k w^j]S(t,u,w)$ counts the s-length segments (0-1-sequences) containing k 1's and having weight j. By *weight* we mean the length of the truncated segment, i.e., the (maybe empty) portion of the segment contained between the leading and trailing 0's. For example, the segment 0010110 has $s = 7, k = 3$ and $j = 4$, whereas 100001 has $s = 6, k = 2$ and $j = 6$. The segment weight is very important because it determines compressed array C's length.

Our problem can be described in the following way: suppose we are given a sparse 0/1 array of length n containing m 1's: we want to find the average weight of the r segments obtained by dividing the array into s-length segments ($n = rs$). We also suppose that s be fixed, and denote the counting generating function describing the set of all s-length segments by $G_s(u,w)$. Function $G_s(u,w)$ can be obtained by expressing $S(t,u,w)$ in partial fractions and then extracting the coefficient of t^s. We have:

$$S(t,u,w) = \frac{u^2w^2}{(1-w-uw)^2}\,\frac{1}{1-t(w+uw)} + \frac{uw(1-w)}{1-w-uw}\,\frac{1}{(1-t)^2} +$$

$$+\frac{1-2w-3uw+w^2+4uw^2+u^2w^2-u^2w^3-uw^3}{(1-w-uw)^2}\,\frac{1}{1-t}.$$

The coefficient of t^s can now be easily obtained because all factors with t correspond to a geometric series:

$$G_s(u,w) = (su^2w^3 + suw^3 - su^2w^2 + u^2w^2(w+uw)^s - 2suw^2 + 2uw^2 + w^2 +$$

$$+suw - 2uw - 2w + 1)/(1-w-uw)^2.$$

We now approach the most important part of our analysis. Since the r segments making up any 0/1 array are independent of each other, their counting generating function is simply

$G_s(u,w)^r$. We can now determine the "total" weight of all the possible r s-long segments (i.e., all possible n-long 0/1 arrays) by simply deriving $G_s(u,w)^r$ with respect to w and by then setting $w=1$. This is a function $J(u)$ only depending on u; since we only treat arrays with m 1's, our last job is to extract the coefficient of u^m in order to obtain the total weight of all the 0/1 arrays having length $n=rs$ and containing exactly m 1's. Let us note that:

$$\left[\frac{d}{dw}G_s(u,w)^r\,\middle|\, w=1\right] = rG_s(u,1)^{r-1}\left[\frac{d}{dw}G_s(u,w)\,\middle|\, w=1\right].$$

This allows us to perform the computations by hand. However, Maple immediately gives us:

$$J(u) = r(1+u)^{rs}\left(\frac{su^2+2u}{u^2(1+u)^s} + 2u^2 + su^2 - 2\frac{1+u}{u}\right) =$$

$$= rs(1+u)^{rs-s} + rs(1+u)^{rs} + \frac{2r}{u}(1+u)^{rs-s} - \frac{2r}{u}(1+u)^{rs}.$$

The coefficient of u^m can now be easily extracted by Newton's rule, and we denote the total weight of all the n-long 0/1 arrays containing exactly m 1's and divided into segments of length s by $K(n,m,s)$:

$$K(n,m,s) = rs\binom{rs-s}{m} + rs\binom{rs}{m} + 2r\binom{rs-s}{m+1} - 2r\binom{rs}{m+1} =$$

$$= n\binom{n-s}{m} + n\binom{n}{m} + \frac{2n}{s}\binom{n-s}{m+1} - \frac{2n}{s}\binom{n}{m+1}.$$

Our arrays' average weight can now be obtained by dividing this amount by the total number of arrays, which is obviously the binomial coefficient $\binom{n}{m}$, i.e., $A(n,m,s) = K(n,m,s)/\binom{n}{m}$. Moreover, the space occupied by the F and Z vectors is $2n/s$ and so we can state the following Theorem:

Theorem 3.1 *The average storage space required by the algorithm described in Section 2 is:*

$$T(n,m,s) = K(n,m,s)\binom{n}{m}^{-1} + \frac{2n}{s},$$

where

$$K(n,m,s) = n\binom{n-s}{m} + n\binom{n}{m} + \frac{2n}{s}\binom{n-s}{m+1} - \frac{2n}{s}\binom{n}{m+1}.$$

■

As a check, we apply this formula to the case illustrated in Figure 3.1, for which $\binom{n}{m} = \binom{6}{2} = 15$. Moreover:

$$K(6,2,3) = 6\binom{3}{2} + 6\binom{6}{2} + 2\frac{6}{3}\binom{6}{3} - 2\frac{6}{3}\binom{6}{3} = 18 + 90 + 4 - 80 = 32$$

and this agrees with the value previously found; finally, we have $A(6,2,3) = 32/15 = 2.133$. For a more realistic example, let $n = 100$ and $m = 10$. If we try to divide the array into segments of length 10 so that $r = 10$ (we get 10 segments and hope that there is a single non-zero element in every segment), we obtain:

$$A(100, 10, 10) \sim 17.48.$$

In other words, in a sparse 100-position array V containing only 10 non-zero elements, the method uses $17 - 18$ memory locations for the compressed array C and 20 other locations for the elements belonging to auxiliary vectors F and Z. We now want to find out if this quantity represents the best we can obtain by this method.

4 Minimizing storage requirements

Since we need a criterion for evaluating the method, we have to determine the value $\bar{s}$ of s minimizing $T(n,m,s)$ (Theorem 3.1) and find out if the corresponding storage $T(n,m,\bar{s})$ can really compete with the space required by other methods (see the Conclusions). By calculus techniques, we could find an appropriate solution to the equation $\frac{d}{ds}T(m,n,s) = 0$ but it is too difficult to perform directly (at least for us). Therefore, we use a relatively simple concept to find out the order of $\bar{s}$ and an approximate solution when some conditions on n, m, s are verified. As we shall see, the most important condition is that $m = o(\sqrt{n})$, i.e., the table is really sparse, as is usually the case. As is intuitively clear, the array V should be divided into a number r of segments not less than m; otherwise, the probability of having more elements (1's) in a single segment becomes too large. However, $r > m$ implies $s = n/r < n/m = c^{-1}$ and this means that $s = o(n)$ or, at least, $s \ll n$. We use the ratio c as a fundamental quantity in the sequel and point out that, if c remains constant, the number $\bar{s}$ should not vary very much, and should approach a limiting value when $n, m \to \infty$. This can be verified in an experimental way by defining a Maple function $T(n,m,s)$ and by then trying various values of s until the minimum is attained.

As previously mentioned, it is not always necessary for s to divide n, and we can experiment with some value ranges to further reduce the $\bar{s}$'s possibilities. When m, n are fixed, $s = n$ implies $T(n,m,s) \sim n$ and $s = 1$ implies $T(n,m,s) \sim 2n$. By considering $T(n,m,s)$ as a continuous function of s, only one minimum exists and some variants of the interval division method quickly allow us to find the $\bar{s}$ desired (see Figure 4.2 for $n = 1000$ and $m = 10$). For example, when $n/m = 100$, we find $\bar{s} = 46$ for $n = 500, m = 5$; $\bar{s} = 45$ for $n = 10000, m = 100$. Analogously, when $n/m = 400$ we find $\bar{s} = 107$ for $n = 10000, m = 25$; $\bar{s} = 106$ for $n = 100000, m = 250$ and $n = 200000, m = 500$.

The use of some Computer Algebra System is now virtually essential. While in the previous section most computations could have been done by hand, in this one they are too complex to be safely done without any mechanical device. We now outline our approach and hope that the reader can follow our indications on his or her computer. It would be interesting to develop an easier, non-mechanical technique and the authors would be grateful for any helpful suggestions on the subject.

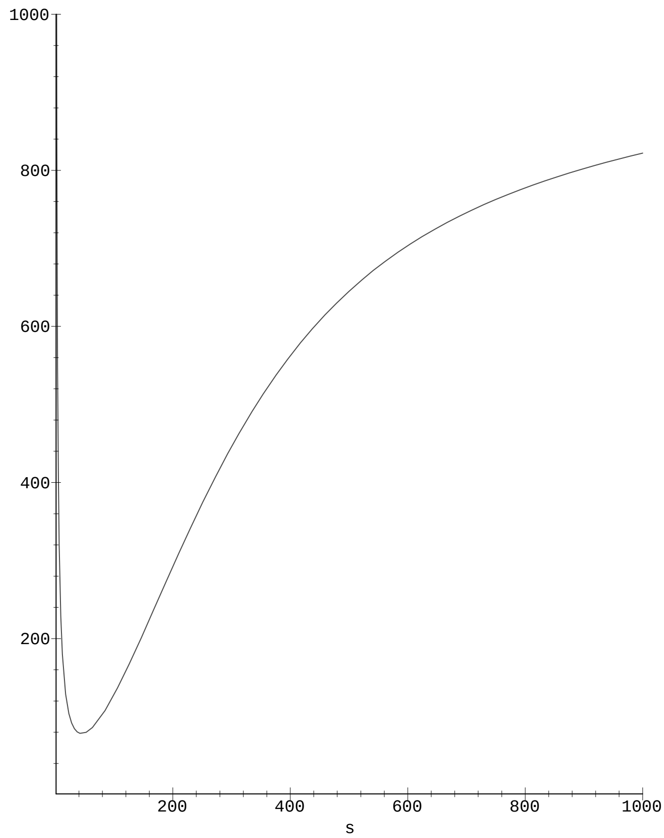

Figure 4.2: The plot of $T(1000, 10, s)$.

Let us begin by defining the quantity:

$$A(n,m,s) = n\binom{n-s}{m} + n\binom{n}{m} + \frac{2n}{s}\binom{n-s}{m+1} - \frac{2n}{s}\binom{n}{m+1} + \frac{2n}{s}\binom{n}{m};$$

this is just $\binom{n}{m}T(n,m,s)$ and the problem of minimizing $T(n,m,s)$ is equivalent to the problem of minimizing $A(n,m,s)$. In order to obtain a more homogeneous expression, we apply the simple rule:

$$\binom{p}{m+1} = \frac{p-m}{m+1}\binom{p}{m}$$

and obtain:

$$A(n,m,s) = n\binom{n-s}{m} + n\binom{n}{m} + \frac{2n(n-m-s)}{s(m+1)}\binom{n-s}{m} - \frac{2n(n-m)}{s(m+1)}\binom{n}{m} + \frac{2n}{s}\binom{n}{m}.$$

By the standard rule:

$$\frac{\Gamma(n+a)}{\Gamma(n+b)} = n^{a-b}\left(1 + \frac{(a-b)(a+b-1)}{2n} + O\left(\frac{1}{n^2}\right)\right)$$

we find:

$$\binom{n}{m} = \frac{\Gamma(n+1)}{m!\Gamma(n-m+1)} = \frac{n^m}{m!}\left(1 - \frac{m(m-1)}{2n} + O\left(\frac{m^4}{n^2}\right)\right) \tag{4.1}$$

$$\binom{n-s}{m} = \frac{\Gamma(n-s+1)}{m!\Gamma(n-m-s+1)} = \frac{(n-s)^m}{m!}\left(1 - \frac{m(m-1)}{2(n-s)} + O\left(\frac{m^4}{(n-s)^2}\right)\right). \tag{4.2}$$

Thanks to our hypotheses $m = o(\sqrt{n})$ and $s = o(n)$, we can eliminate the O-terms in these expressions and substitute them in $A(n,m,s)$. The quantities $m!$ amd $(m+1)$ can be eliminated from the denominator because they do not depend on s; Maple eventually finds the following expression:

$$\begin{aligned}
&\frac{1}{2}(4\,n^m\,m^3\,n + 2\,n^m\,s\,m^2 - 2\,n^m\,n^2\,m^2 - 6\,n^m\,n^2\,m - 6\,n^m\,s\,n^2 + 2\,n\,(n-s)^m\,m^2\\
&\quad - 2\,n\,(n-s)^m\,s^2 + 6\,n^2\,(n-s)^m\,s + 4\,n^m\,s\,n - 4\,n^m\,n^2 - 4\,n^3\,(n-s)^m + 4\,n^m\,n^3\\
&\quad + (n-s)^m\,s\,m^3\,n + 2\,n^m\,s\,m + 2\,(n-s)^m\,n^2\,m - 2\,(n-s)^m\,n\,m^3 - 3\,(n-s)^m\,s\,n\,m\\
&\quad - 2\,(n-s)^m\,s\,m^2\,n + 5\,n^m\,s\,n\,m + 2\,n\,(n-s)^m\,s^2\,m - 2\,n^2\,(n-s)^m\,s\,m + 2\,n^m\,s^2\,n\,m\\
&\quad - 2\,n^m\,s\,n^2\,m + 2\,n^m\,s\,m^2\,n + n^m\,s\,m^3\,n - 2\,n^m\,n\,m^2 + 2\,n^m\,s^2\,n - 4\,n^m\,s\,m^3 - 2\,n^m\,n\,m\\
&\quad + 2\,(n-s)^m\,n^2\,m^2 + n^m\,s^2\,m - n^m\,s^2\,m^3)/(s\,(-n+s))
\end{aligned}$$

which can be differentiated with respect to s. The resulting expression is now equated to 0 and we can try to find the root corresponding to the value $\bar{s}$. By dividing everything by n^m, we obtain the following expression:

$$\begin{aligned}
&\frac{1}{2}n(6\,s\,m^3 - 4\,s\,n^2\,m + m^4\,s^2 - 2\,m^4\,s + 6\,s^2\,n\,m - 4\,s\,n\,m + 2\,s\,m^3\,n - s^2\,m^2 + 3\,s^2\,m\\
&\quad - 2\,s\,m^2\,n - 3\,m^3\,s^2 - 4\,n\,s^2 - 2\,m^3\,n + 2\,m^2\,n + 2\,n^2\,m + 8\,n^2\,s + 2\,n^2\,m^2 - 2\,s^2\,m^2\,n\\
&\quad - 2\,s^3\,m - 4\,s\,m^2 - 4\,n^3 + 2\,s^3\,m^2)\left(\frac{n-s}{n}\right)^m/((-n+s)^2\,s^2)\\
&\quad + \frac{(n-1-2\,m)\,(2\,n+m-m^2)}{s^2}
\end{aligned}$$

At this point, we wish to make two basic observations:

1. because of our hypotheses $m = o(\sqrt{n})$ and $s = o(n)$, the quantity $((n-s)/n)^m$ is smaller than 1, and we have:

$$\left(\frac{n-s}{n}\right)^m = \left(1 - \frac{s}{n}\right)^m < \left(1 - \frac{s}{n}\right)^{n/s} \sim e^{-1}.$$

   Let us denote it by y; since it is a limited quantity, it cannot influence the behaviour of the above expression when $n, m \to \infty$;

2. by setting $c = m/n$ and letting $m, n \to \infty$, some terms' growth order is faster than others. Since only the terms of maximal growth influence the value of $\bar{s}$, the others can be ignored. This can be seen more clearly, if we divide everything by the largest powers of m, n and eliminate the terms that approach 0 most rapidly.

We go on to substitute y for $((n-s)/n)^m$ and cn for m, and take the numerator divided by n. We get the following expression:

$$
\begin{aligned}
&c^2\,(y\,s^2\,c^2 - 2\,y\,s\,c^2 + 2\,y\,s\,c - 2\,y\,c + 4\,c - 2 + 2\,y)\,n^4 - (2\,y\,s^2\,c^2 + 2\,y\,s\,c^2 \\
&\quad + 4\,y\,s\,c - 2\,y\,c^2 - 2\,y\,c + 8\,c^3\,s + 3\,y\,c^3\,s^2 + 2\,c^2 + 6\,c - 4 - 6\,y\,s\,c^3 - 4\,s\,c^2 + 4\,y)n^3 + \\
&\quad (2\,y\,s^3\,c^2 + 12\,c\,s - 8\,s - 4\,y\,s\,c^2 - 2\,c^2\,s^2 - 4\,y\,s\,c + 4\,s\,c^2 + 4\,s^2\,c^3 + 6\,y\,s^2\,c + 8\,y\,s \\
&\quad - y\,s^2\,c^2 - 4 - 2\,c)n^2 - \ s\,(2\,y\,s^2\,c + 6\,c\,s - 4\,s + 2\,s\,c^2 + 4\,y\,s - 3\,y\,s\,c - 8 - 4\,c)\,n \\
&\quad - 4\,s^2 - 2s^2\,c
\end{aligned}
$$

By observation 2. above, we only take into account the coefficient of n^4. Now, we eliminate the y as follows:

$$y = \left(1 - \frac{s}{n}\right)^m \sim 1 - \frac{ms}{n} + \binom{m}{2}\frac{s^2}{n^2} - \binom{m}{3}\frac{s^3}{n^3}.$$

It is worth noting that after developing this expression, the terms having $degree(m) < degree(n)$ are to be eliminated again because of observation 2. Consequently, we could substitute:

$$y = 1 - \frac{ms}{n} + \frac{m^2 s^2}{2n^2} - \frac{m^3 s^3}{6n^3} = 1 - cs + \frac{c^2 s^2}{2} - \frac{c^3 s^3}{6}$$

and obtain the same results. At any rate, after some simplifying, we eventually obtain the equation desired:

$$-c^4\,s^5 + c^3\,(1 + 2\,c)\,s^4 - 2\,c^2\,(1 + 2\,c)\,s^3 + 6\,c^2\,s^2 + 12 = 0.$$

Our final task is to solve this equation, or at least to find an approximate solution able to give us the $\bar{s}$'s order. This can be done by the Newton-Raphson method if we are able to guess a first rough approximation of the actual root. This is relatively easy if we consider the $\bar{s}$ values obtained at the beginning of this section and/or if we try to solve our equation numerically for different values of c. The results are the same and we optimistically assume $\sqrt[3]{n^2/m^2} < \bar{s} < 2\sqrt[3]{n^2/m^2}$. Therefore, we set $s_0 = 2/c^{2/3}$ and apply the Newton-Raphson method obtaining:

$$s_{i+1} = s_i - \frac{c^4\,s_i^5 - c^3\,(1 + 2\,c)\,s_i^4 + 2\,c^2\,(1 + 2\,c)\,s_i^3 - 6\,c^2\,s_i^2 - 12}{5\,c^4\,s_i^4 - 4\,c^3\,(1 + 2\,c)\,s_i^3 + 6\,c^2\,(1 + 2\,c)\,s_i^2 - 12\,s_i\,c^2} \tag{4.3}$$

By substituting s_0 in this expression and by then developing the result into a Laurent series, we find:

$$s_1 := \frac{11}{6}\frac{1}{c^{2/3}} + \frac{4}{9}\frac{1}{c^{1/3}} + \frac{35}{54} + \cdots = \frac{1.833\ldots}{c^{2/3}} + \frac{.444\ldots}{c^{1/3}} + .648148148\ldots + \cdots$$

We go along in the same way and find:

$$s_2 := \frac{1979}{1089}\frac{1}{c^{2/3}} + \frac{472933}{862488}\frac{1}{c^{1/3}} + \frac{21797783}{42693156} + \cdots =$$

$$= \frac{1.817263\ldots}{c^{2/3}} + \frac{.548335\ldots}{c^{1/3}} + .510566\cdots$$

$$s_3 := \frac{23250081292}{12795012747}\frac{1}{c^{2/3}} + \frac{182100709636079819}{330899143397458284}\frac{1}{c^{1/3}} + \frac{855764291061707443316957 3}{1711514404342338405732489 6} + \cdots =$$

$$= \frac{1.8171206\ldots}{c^{2/3}} + \frac{.550320\ldots}{c^{1/3}} + .500\cdots$$

It is now not too difficult to guess that the three constants are $\sqrt[3]{6}$, $1/\sqrt[3]{6}$ and $1/2$ respectively, and therefore we set:

$$s = \sqrt[3]{\frac{6}{c^2}} + \sqrt[3]{\frac{1}{6c}} + \frac{1}{2} + O(c)$$

and insert this value into (4.3). By developing the result into a Laurent series, we find the same value and thus show that this is both a fixed point for (4.3) and a solution to our equation. We conclude with the following result, which is the central point of our paper:

Theorem 4.1 *Given n and $m = o(\sqrt{n})$, the value of s minimizing the space occupied by the method when $n, m \to \infty$ while the ratio n/m remains constant is:*

$$\bar{s} = \sqrt[3]{\frac{6n^2}{m^2}} + \sqrt[3]{\frac{n}{6m}} + \frac{1}{2} + O(c).$$

■

With regard to the examples given at the beginning of this section, when $n/m = 100$, the value of $\bar{s}$ is 42.2; when $n/m = 400$, the value of $\bar{s}$ is 103.2. This empirically shows that our analysis actually agrees with real cases.

We now use the value for $\bar{s}$ above to find an asymptotic expression for the average space occupied by the compression method. First of all, we compute the asymptotic value for $\binom{n-s}{m}/\binom{n}{m}$. By the expressions (4.1), (4.2), we have:

$$y = \binom{n-s}{m}\bigg/\binom{n}{m} \sim \left(1 - \frac{s}{n}\right)^m \left(1 - \frac{m(m-1)s}{n(n-s)}\right) \sim \left(1 - \frac{sc}{m}\right)^m \left(1 - \frac{c^2 s}{2}\right)$$

By developing this formula we find:

$$y \sim 1 - \sqrt[3]{6c} + \sqrt[3]{\frac{4c^2}{3}} - \frac{c}{2} + O(c^{4/3}).$$

The storage occupied by the compressed array C is:

$$T = n\left(y + 1 + 2\frac{(n-m-s)y}{s(m+1)} - 2\frac{n-m}{s(m+1)}\right).$$

We now substitute nc to m and let $n \to \infty$:

$$\frac{T}{n} \sim \frac{ycs + cs + 2y - 2cy - 2 + 2c}{cs}.$$

Therefore the storage for C is:

$$T \sim n\left(\sqrt[3]{\frac{c^2}{6}} + \frac{5}{6}c + O(c^{4/3})\right).$$

For what concerns the arrays F and Z, we have:

$$F = \frac{2n}{s} \sim n\left(2\sqrt[3]{\frac{c^2}{6}} - \frac{1}{3}c + O(c^{4/3})\right).$$

Therefore, the average space occupied by the method is given by the following:

Theorem 4.2 *The minimal average storage space occupied by the compression method of Section 2 is:*

$$A = A(n,m) = 3\sqrt[3]{m^2 n} + \frac{m}{2} + O\left(m\sqrt[3]{\frac{m}{n}}\right).$$

Proof: By summing $T + F = A$ and substituting $m/n = c$, we immediately find the formula desired. ∎

5 Compression in $O(m^{4/3})$ space

The average storage space occupied by the compression method described grows like $\sqrt[3]{nm^2}$, and therefore it increases too much when n is large, as it usually is. We now want to find if there is a variation of the method allowing us to minimize the storage required and maintain it in the order of (some small power of) m. The simple algorithm we propose can be roughly described as follows:

- we find a suitable $p < n$ for which the mapping $\phi_p(x) = x \bmod p$ is injective on S;
- we apply the compression method to the universe $[1..p]$ and set $S' = \{x' | x' = (x \bmod p) + 1, x \in S\}$ (we add 1 to $x \bmod p$ in order to get $1 \leq x' \leq p$, as required by the algorithm described in Section 2).

If we had $p = O(m)$, the total storage would be $O(m)$, and since the operation "$x \bmod p$" only takes constant time, we should obtain an optimal compression technique. Unfortunately, this is not so and we can only obtain $p = O(m^2)$, so the total space occupied is $O(m^{4/3})$ by Theorem 4.2. This is not a bad result and can easily be achieved in $O(m)$ time as follows.

Let $V = [1..n]$, $S \subseteq V$ with $|S| = m$ and $m \leq p \leq n$; we begin by determining the probability $\pi(n,m,p)$ that the mapping $\phi_p(x) = x \bmod p$ is injective on S :

Theorem 5.1 *Given n, m, and p as indicated above, then the probability that the mapping $\phi_p(x) = x \bmod p$ is injective on S is:*

$$\pi(n,m,p) \approx \binom{p}{m}\left(\frac{n}{p}\right)^m \Big/ \binom{n}{m}$$

Proof: Let us assume that p divides n, so that $n = pk$ for some $k \in \mathbf{N}$; the congruence $x \equiv y (\mathrm{mod}\ p)$ divides V into p different equivalence classes, all having exactly k elements. A subset $S \subseteq V$ is formed by taking an element out of m different equivalence classes, i.e., in $\binom{p}{m}$ different ways. The classes are disjoint pairwise, and therefore the elements are extracted independently. This means that the possible number of choices is $k^m = (n/p)^m$ and so the total number of subsets S is $\binom{p}{m}(n/p)^m$. Since the total number of subsets is $\binom{n}{m}$, the theorem is proved in this particular case. In general, n/p is the average number of elements per class, and every class contains $\lfloor n/p \rfloor$ or $\lceil n/p \rceil$ elements; this explains why we have "$\approx$" instead of "$=$"in the theorem's statement. ∎

Let us now fix a probability α $(0 < \alpha < 1)$ and determine a value of p for which $\pi(n, m, p) \geq \alpha$. We can try to reduce S modulo p by the mapping ϕ_p and verify if ϕ_p is injective; if so, we have finished. Otherwise, we go on to try with $\phi_{p-1}, \phi_{p+1}, \phi_{p-2}, \phi_{p+2}$, and so on, until we find that the mapping is injective on S. We can consider each ϕ as being independent from the others; therefore, after k trials, the probability of finding a correct ϕ is greater than $1 - (1 - \alpha)^k$, i.e., very close to 1.

It is not very easy to find p so that $\pi(n, m, p) \approx \alpha$, but we can find an approximation of $\pi(n, m, p)$ when $n \to \infty$, and this simplifies the problem. We give two proofs of the following:

Theorem 5.2 *When $n \to \infty$, we have:*

$$\pi(m, p) = \lim_{n\to\infty} \pi(n, m, p) = \lim_{n\to\infty} \binom{p}{m}\left(\frac{n}{p}\right)^m \bigg/ \binom{n}{m} = \frac{p^{\underline{m}}}{p^m} = \frac{p!}{(p-m)!p^m},$$

where $p^{\underline{m}} = p(p-1)\dots(p-m+1)$.

Proof: The first proof is algebraic. We have:

$$\binom{p}{m}\left(\frac{n}{p}\right)^m \bigg/ \binom{n}{m} = \frac{p!}{(p-m)!p^m}\frac{n^m}{p^m}\frac{m!(n-m)!}{n!} = \frac{p^{\underline{m}}}{p^m}\frac{n^m(n-m)!}{n!}.$$

Let us assume that m is fixed as $n \to \infty$. By Stirling's approximation, we obtain:

$$\lim_{n\to\infty}\frac{n^m(n-m)!}{n!} = \lim_{n\to\infty}\sqrt{\frac{n-m}{n}}\frac{(n-m)^{n-m}}{n^n}n^m e^m = \lim_{n\to\infty}\sqrt{\frac{n-m}{n}}\left(1-\frac{m}{n}\right)^{n-m} e^m.$$

This shows that the limit value is 1 and the theorem is proven. The second proof is combinatorial: when $n \to \infty$, we can assume that the mapping ϕ_p acts as a random transformation from $[1..n]$ into $[0..p-1]$. Therefore, the problem is equivalent to inserting m balls into p different urns in such a way that each urn contains at most one ball. The corresponding probability is obviously:

$$\pi(m, p) = \frac{p}{p}\frac{p-1}{p}\frac{p-2}{p}\dots\frac{p-m+1}{p} = \frac{p^{\underline{m}}}{p^m}.$$

∎

By using this limiting approximation $\pi(m, p)$, we find p as a function of m and α :

Theorem 5.3 *The value of p for which the probability $\pi(m,p)$ is α is:*

$$p \approx \frac{m^2}{\ln 1/\alpha^2} - \frac{m}{\ln 1/\alpha^2} + \frac{m}{3} + O(1).$$

Proof: By applying Stirling's approximation, we get:

$$\frac{p^{\underline{m}}}{p^m} = \frac{p!}{(p-m)!p^m} \sim \frac{\sqrt{2\pi p}(p/e)^p}{\sqrt{2\pi(p-m)}((p-m)/e)^{p-m}p^m} = \sqrt{\frac{p}{p-m}}\frac{p^{p-m}}{(p-m)^{p-m}e^m}.$$

By equating this to α, we have the equation:

$$\frac{p^{p-m+1/2}}{(p-m)^{p-m+1/2}} = \alpha e^m,$$

and by evaluating the logarithms, we obtain:

$$\left(p-m+\frac{1}{2}\right)\ln\left(\frac{p}{p-m}\right) = m + \ln\alpha.$$

If we assume that $m \ll p$, we can now develop the logarithm into a power series:

$$\left(p-m+\frac{1}{2}\right)\left(\frac{m}{p}+\frac{m^2}{2p^2}+\frac{m^3}{3p^3}+\cdots\right) = m+\ln\alpha.$$

Since the powers of m/p become smaller and smaller, we can stop our development at the second power and obtain a second degree equation:

$$12p^2\ln 1/\alpha - 6m(m-1)p - m^2(2m-3) = 0.$$

Only one of the two solutions is positive:

$$p = \frac{3m(m-1)+3m^2\sqrt{1-\left(2-\frac{8}{3}\ln\frac{1}{\alpha}\right)\frac{1}{m}+\left(1-4\ln\frac{1}{\alpha}\right)\frac{1}{m^2}}}{12\ln\frac{1}{\alpha}}.$$

By now developing the square root into a power series, we obtain the results desired. We could improve our formula by finding some other terms, but the formula used is sufficient for our purposes. ∎

We can fix $\alpha = 1/2$ so that $\ln 1/\alpha^2 = \ln 4$, and state our algorithm more precisely:

- Let the set $S = \{x_1, x_2, \ldots, x_m\} \subseteq [1..n]$ be given.
- Set $\hat{p} := \frac{m^2}{\ln 4} - \frac{m}{\ln 4} + \frac{m}{3}$ and $j := 0$;
- <u>Repeat</u>: $p = \hat{p} + j$;
  T:=$\emptyset$;
  <u>for</u> $i = 1..m$ $T_i := (S_i \bmod p) + 1; T = T \cup \{T_i\}$ <u>endfor</u>
  $v := |T| = |S|$; <u>if</u> <u>not</u> v <u>then</u> $j := (|j|+1)(-1)^{|j|+1}$ <u>endif</u>
  <u>until</u> <u>not</u> v.

- Apply the compression algorithm described in Section 2 to T.

We can now solve the static dictionary problem in the following way: given an element $x \in V$, we reduce it modulo p, $x' = (x \bmod p) + 1$, and then we search x' in the compressed array C (obtained from T) as described in Section 2. If $x' \in C$, then $x \in S$, and, if $x' \notin C$, then $x \notin S$. The other problems are solved accordingly.

Our final result concerns the evaluation of the space occupied by the new algorithm. Strictly speaking, we have $m = O(\sqrt{p})$ and not $m = o(\sqrt{p})$ (required by the results in Section 4). However, we can apply the result in Theorem 4.2 to get a rough idea of the amount of storage required by the previous algorithm:

Corollary 5.4 *The space occupied by the previous algorithm is in the order of* $O(m^{4/3})$.

∎

We wish to conclude our paper with a simple but meaningful example. Let:

$$S = \{JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC\}$$

and consider the ASCII equivalent of these strings when stored in three bytes. We get the following integer values:

$$S = \{4866382, 4605250, 5062994, 4280402, 5063001, 4871502,$$

$$4871500, 4281671, 5457232, 5194580, 5132118, 4474179\}.$$

If we apply the formula in Theorem 5.3 to $m = 12$ (being $n = 2^{24}$), we find $p = 99.21$. By performing the computations by hand, we hopefully begin with $p = 100$, but $APR = 4280402$ and $JUN = 4871502$ give the same result. We then try setting $p = 99$ and we successfully find:

$$T = \{38, 68, 36, 39, 43, 10, 8, 21, 56, 51, 58, 73\}.$$

If we sort these elements, we find our first example in Section 2: the compressed array C contains 16 elements and the two vectors F and Z both occupy 11 memory locations. The total storage is 38 locations, or words, and this result is not too bad.

6 Conclusions

In this paper, we do not aim at presenting an optimal method of solving the static dictionary or the sparse matrix compression problems but rather, we want to show how an accurate analysis can suggest a substantial improvement of a method. On the other hand, a space occupancy in the order of $O(m^{4/3})$ for a size m table is not particularly satisfactory and some better techniques have been proposed. For the static dictionary problem, we mention the paper [?], in which optimum storage utilization $O(m)$ is attained, at the cost of adding $O(m)$ information to the basic minimal perfect hashing table (for optimality bounds, see [4]).

As far as our method is concerned, we can improve space occupancy by performing a "segmentation" of the set X. We can do this easily by using a modulo v operation, after dividing X into v subsets, each of size $\approx m/v$. We then apply our method to each segment and the information needed for each is stored in $O(v)$ space. Therefore, if $v = m^u$ for some u, $0 < u < 1$, each segment occupies $K_1 m^{4(1-u)/3}$ storage locations, for some constant K_1. Occupancy is therefore $K_1 m^u m^{4(1-u)/3} + K_2 m^u$ in all and the actual problem is to find the value of u that minimizes this quantity. However, this depends on the two constants K_1 and K_2, which are obviously implementation sensitive. Consequently, though not difficult, the problem goes beyond the scope of this paper. For example, if $K_1 = K_2$, we have $u = 1 - \frac{3}{4}\log_m 3$; for m large, $u \approx 1$, and this indicates that the segments have to be small. This is rather intuitive, and using a practical method would involve choosing v so that $m/v \approx 10 \div 12$. In any case, the preprocessing time remains $O(m)$.

Aknowledgments

We wish to thank the anonymous referee whose comments helped us in improving the contents and readability of our paper.

References

[1] A. Aho, R. Sethi, and J. Ullman. *Compilers: Principles, techniques and tools.* Addison Wesley, 1988.

[2] A. Gibbons, S. Muthukrishnan, and I. Pu. Exact expected case analysis of a simple heuristic for compressed array storage. Unpublished.

[3] I. P. Goulden and D. M. Jackson. *Combinatorial Enumeration.* John Wiley & S., 1983.

[4] K. Mehlhorn. *Data structures and algorithms 1: sorting and searching.* Spinger Verlag, 1984.

[5] M. P. Schützenberger. Context-free language and pushdown automata. *Information and Control*, 6:246–264, 1963.

[6] R. Sprugnoli. Perfect hash functions: a single probe retrieving method for static sets. *Communications of the ACM*, 20(11):841–850, 1977.

[7] R. Tarjan. Graph theory and Gaussian elimination, 1976. In Sparse Matrix Computations, J. Bunch and D. Rose, Academic Press.