Data Compression and Gray-code Sorting

by

Dana Richards

Computer Science Report #TR-85-24

November 1985

Submitted for publication in Information Processing Letters.

Data Compression and Gray-code Sorting

Dana Richards
University of Virginia

Introduction

Data compression techniques have been studied many years. We are concerned with file compression when all the records of the file have the same field structure. In particular we assume each record has m fields, $f_1, f_2, \cdots, f_m$. The field f_i has integer values in the range 0 to N_i-1, so that for a record $X = (x_1, x_2, \cdots, x_m)$, we have $0 \leqslant x_i \leqslant N_i-1$, for $1 \leqslant j \leqslant m$.

The compression scheme we analyze here, known as <u>differencing</u> ,was discussed by Ernvall [ERNV84]. The idea is to arrange the records in some order and output the first record. For each successive record we output only those fields which differ from the previous record. If most successive records have many identical fields then the output file should be much shorter than the original file. Note that there is some overhead for specifying which field is being output and end-of-record delimiters, so that output file could be longer. (If we were allowed to reference any record, not just the previous one, then a non-linear structure develops and leads to the use of minimum spanning trees [ERNV79, KANG77].)

The initial ordering of the records is important and Ernvall chooses a generalized Gray code order discussed below. To motivate this consider the case when all the $N_i = 2$ and the file consists of the 2^m possible records, i.e. every m-bit sequence. If these where arranged in Gray code order, by definition, each successive record would differ in exactly one field giving $m + 2^m-1$ output fields, as opposed to the $m2^m$ of the naive method. If we had simply put the records in lexicographic order then the number of output fields would be $m + 2^m \sum_{i=0}^{m-1} \frac{i}{2^i} = 2^{m+1}-m-2$ which asymptotically is just twice as bad. However in the worst case we could have an ordering such that each successive record differs in

$m-1$ fields [ROBI81] giving an unfavorable $2^m(m-1)+1$ output fields.

The problem Ernvall explored is how to sort the records initially so that they are in a Gray-code order. This does not mean that each successive record differs in just one field. Instead it means the records that do appear in the file are in the same order as they appear in a Gray-code ordering of all potential records, over the ranges specified. We present an improvement to Ernvall's algorithm and extend it to the full mixed-radix case and present another algorithm. In the next section we give definitions and some results which are interesting by themselves.

Preliminaries

Let S_1 be an ordered sequence of n_1 elements, denoted $[S_1(0), S_1(1), \cdots, S_1(n_1-1)]$. Let $S_1(i)S_2(j)$ represents an ordered pair formed by concatenating the respective elements of two sequences; and $S_1(i)S_2$ is the sequence $[S_1(i)S_2(0), S_1(i)S_2(1), \cdots, S_1(i)S_2(n_2-1)]$. Let $S_1 \times S_2 = [S_1(0)S_2, S_1(1)S_2, S_1(2)S_2, \cdots, S_1(n_1-1)S_2]$ be a sequence of $n_1 n_2$ ordered pairs where all the elements of one subsequence proceeds the next subsequence. Finally $S_1 \otimes S_2 = [S_1(0)S_2, S_1(1)S_2^R, S_1(2)S_2, S_1(3)S_2^R, \cdots, S_1(n_1-1)S_2^{(R)}]$ where $S_2^{(R)}$ denotes either S_2 or S_2^R, i.e. the reversal of S_2, depending on whether n_1-1 is even or odd, respectively. As an example let $S_1 = [0,1,2]$ and $S_2 = [A, B]$, then $S_1 \otimes S_2 = [0A, 0B, 1B, 1A, 2A, 2B]$. Further $S_1 \otimes (S_2 \otimes S_1)$ is, arranged in two columns:

0A0	1A2
0A1	1A1
0A2	1A0
0B2	2A0
0B1	2A1
0B0	2A2
1B0	2B2
1B1	2B1
1B2	2B0

It is easy to derive rules for indexing into these compound sequences. We see $S_1 \times S_2(i) = S_1(i_1)S_2(i_2)$, where $i_1 = \left\lfloor \frac{i}{n_2} \right\rfloor$ and $i_2 = i \bmod n_2$. Similarly, $S_1 \otimes S_2(i) = S_1(i_1)S_2(i_2)$, where $i_1 = \left\lfloor \frac{i}{n_2} \right\rfloor$ and $i_2 = i \bmod n_2$ or $i_2 = n_2-1-(i \bmod n_2)$ when i_1 is even or odd, respectively. For example, with $n_1 = 3$, $n_2 = 2$, and S_1 and S_2 as above, $S_1 \otimes (S_2 \otimes S_1)(8) = S_1(1)S_2 \otimes S_1(2\times3-1-2) = S_1(1)S_2(1)S_1(3-1-0) = 1B2$.

The example of $S_1 \otimes (S_2 \otimes S_1)$ produced a sequence of ordered triples that is a Gray code sequence, i.e. each differs from the preceding in exactly one position. We can use this same approach to generate a Gray code sequence of all possible records with m fields, as described in the preceding section. Recall N_a is the number of values in field f_a and let $N_a^b = N_a N_{a+1} \cdots N_b$, the number of possible subrecords with the contiguous fields $f_a, \cdots, f_b$. Let G_a^b be a Gray code sequence for those N_a^b subrecords, where G_a^a is just the ordered sequence $[0, 1, \cdots, N_a - 1]$. We want G_1^m. The following recursive definition is a simple generalization of the well-known binary reflected Gray code (e.g. [BITN76, GILB58]). Let

$$G_a^b = G_a^a \otimes G_{a+1}^b.$$

A simple induction argument establish that each of the N_a^b possible records appear exactly once and it is, in fact, a Gray code sequence. To generate the lexicographically ordered sequence of the same records we use the straightforward definition $L_a^b = L_a^a \times L_{a+1}^b$, where $L_a^a = G_a^a$.

We wish to establish an algebraic property of these sequences. We begin with this result for arbitrary sequences S_1, S_2, S_2 of lengths n_1, n_2, n_3.

Lemma $(S_1 \otimes S_2) \otimes S_3 = S_1 \otimes (S_2 \otimes S_3)$

Proof: Let i be the mixed-radix [REIN77] number $<i_1 i_2 i_3>$ with radices n_1, n_2, n_3, i.e. $i = i_1 n_2 n_3 + i_2 n_3 + i_3$, where $0 \leqslant i_j < n_j$ for $j = 1, 2, 3$. Using the indexing scheme above we find $(S_1 \otimes S_2) \otimes S_3(i) = S_1 \otimes S_2(a')S_3(a_3) = S_1(a_1)S_2(a_2)S_3(a_3)$ where

$$a' = \left\lfloor \frac{i}{n_3} \right\rfloor = i_1 n_2 + i_2$$

$$a_3 = \begin{cases} i \mod n_2 = i_3 & a' \text{ even} \\ n_3 - 1 - (i \mod n_{3)} = n_2 - 1 - i_3 & a' \text{ odd} \end{cases}$$

so

$$a_1 = \left\lfloor \frac{a'}{n_2} \right\rfloor = i_1$$

$$a_2 = \begin{cases} a' \mod n_2 = i_2 & i_1 \text{ even} \\ n_2-1-(a' \mod n_2) = n_2-1-i_2 & i_1 \text{ odd.} \end{cases}$$

Similarly $S_1 \otimes (S_2 \otimes S_3)(i) = S_1(b_1)S_2 \otimes S_3(b') = S_1(b_1)S_2(b_2)S_3(b_3)$ where

$$b_1 = \left\lfloor \frac{i}{n_1 n_2} \right\rfloor = i_1$$

$$b' = \begin{cases} i \mod n_1 n_2 = i_2 n_2 + i_3 & b_1 \text{ even} \\ n_2 n_3-1-(i \mod n_2 n_3) & b_1 \text{ odd} \end{cases}$$

so

$$b_2 = \left\lfloor \frac{b'}{n_3} \right\rfloor = \begin{cases} i_2 & b_1 \text{ even} \\ n_2-1-i_2 & b_1 \text{ odd} \end{cases}$$

$$b_3 = b' \mod n_3 = \begin{cases} i_3 & b_1 \text{ even}, b_2 \text{ even} \\ n_3-1-i_3 & b_1 \text{ odd}, b_2 \text{ even} \end{cases}$$

$$b_3 = n_3-1-(b' \mod n_3) = \begin{cases} n_3-1-i_3 & b_1 \text{ even}, b_2 \text{ odd} \\ i_3 & b_1 \text{ odd}, b_2 \text{ odd} \end{cases}$$

By a case analysis we find, in all cases, $a_1 = b_1$, $a_2 = b_2$, and $a_3 = b_3$. □

<u>Theorem</u> $G_a^c = G_a^b \otimes G_{b+1}^c$, $a \leq b \leq c$.

<u>Proof:</u> A simple induction proof, on $k = c-a+1$, can be made using the observation.

$$G_a^b \otimes G_{b+1}^c = G_a^b \otimes G_{b+1}^{b+1} \otimes G_{b+2}^c = G_a^{b+1} \otimes G_{b+2}^c$$

where $b+1 < c$. □

This theorem allows us to approach the Gray code by decomposing it in other ways than the typical left-to-right fashion. In a similar but simpler manner we find $(S_1 \times S_2) \times S_3 = S_1 \times (S_2 \times S_3)$ and $L_a^c = L_a^b \times L_{b+1}^c$, $a \leq b \leq c$.

In the next section we will need to know the parity of the rank of a record X in the Gray code order. The following result generalizes a result in [ERNV84].

<u>Lemma</u> If $X = G_a^b(i) = (x_a, \cdots, x_b)$ then

$$i \equiv \sum_{j=a}^{b} x_j \mod 2.$$

Proof We prove it by induction on $k = c-a+1$. Recall $X = G_a^{b-1} \otimes G_b^b(i) = G_a^{b-1}(i_1)G_b^b(i_2)$, where $i_1 = \left\lfloor \frac{i}{N_b} \right\rfloor$, $i_2 = i \mod N_b$ or $i_2 = N_b-1-(i \mod N_b)$ as i_1 is even or odd. Note that $x_b = G_b^b(i_2) = i_2$ and $(i \mod N_b) \equiv (i_1-1)i_2 + (i_2-N_b+1)i_1 \mod 2$. Hence

$$
\begin{aligned}
i &\equiv i_1 N_b + (i \mod N_b) \mod 2 \\
&\equiv i_1 - i_2 \mod 2 \\
&\equiv \left(\sum_{j=1}^{b-1} x_j\right) + x_b \mod 2 . \ \square
\end{aligned}
$$

It is interesting that this result does not extend to the lexicographic order, i.e. for the record $Y = L_a^b(i) = (y_a, \cdots, y_b)$. It is easy to show that, in this case, i is equal to the mixed radix number $<y_a \cdots y_b>$ with the radices $N_a, N_{a+1}, \cdots, N_b$. If desired, the multiplications required to calculate i can be bypassed by using a series of parity checks, in the Horner's rule fashion to calculate the parity of i. Further the parity calculation is trivial when all radices are even.

Sorting with Generalized Comparisons

Recall our goal is to sort the file in Gray code order, i.e. record X precedes record Y iff X precedes Y in G_1^m. Ernvall [ERNV84] used the approach of simply employing some known good sorting algorithm and augmenting the notion of a comparison. In particular, he proposes using a boolean function *grayorder*(X, Y) for <u>every</u> comparison step, which returns whether X precedes Y. When $X \neq Y$ let d be the least integer such that $x_{d+1} \neq y_{d+1}$ and let $total_d = \sum_{j=1}^{d} x_i$. He gave the following procedure, which we have generalized to the mixed-radix case,

grayorder (X, Y) = **if** $total_d$ is even **then**

return$(x_{d+1} < y_{d+1})$

else

return$(y_{d+1} < x_{d+1})$

This follows because both X and Y are in the same subsequence $G_1^d(i_1)G_{d+1}^{m(R)}$, where the (R) indicates reversal iff i_1 odd. Recall $i_1 \equiv total_d \bmod 2$. Since $G_{d+1}^m = G_{d+1}^{d+1} \otimes G_{d+2}^m$ it follows that ranking within G_{d+1}^{d+1} determines the ranking in G_{d+1}^m and the result follows.

Notice that every comparison requires $O(m)$ time. Hence if an optimal $O(n \log n)$ comparison-based sorting algorithm is used, as Ernvall recommends, we use $O(m\, n \log n)$ time. However if we, during preprocessing, calculate the rank of each record in Gray-code order, we could then simply sort according to rank. If each rank can be compared in $O(1)$ time and can be computed in $O(m)$ time then the new algorithm would run in $O(nm + n \log n)$ time.

To compute the rank of subrecord X, i.e. i where $X = G_a^c(i)$, we note $X = G_a^b(i_1)G_{b+1}^c(i_2)$ and $i = i_1 N_{b+1}^c + i_2'$, where $i_2' = i_2$ or $i_2' = N_{b+1}^c - 1 - i_2$ as i_1 is even or odd. So if we, perhaps recursively, calculate i_1 and i_2 we can return i. The simplest $O(m)$ implementation of this scheme would involve a left-to-right scan, in the typical Horner's rule fashion;

```
i ← x_1
for i ← 2 to m do
    i_2 ← if i even then x_j else N_j−1−x_j
    i ← iN_j + i_2
endfor
```

However, our general decomposition approach could easily be implemented by a recursive $O(m)$ time approach, with depth $O(\log m)$ when the division is by half at each step. Further if file compression became a bottleneck in a large file system then it may be economical to design special-purpose hardware to compute ranks. The observations above lead naturally to an $O(\log m)$ time parallel implementation.

If some system is already designed to sort in lexicographic order then we can achieve the same end by filtering the file before and after sorting. In particular, record $X = G_1^n(i)$ is transformed into $Y = L_1^n(i)$ before the sort and the reverse mapping is done after sorting. Clearly the result is in Gray-code order. Using the techniques above all the transformations can be done in $O(nm)$ time. (Such transformations with the binary reflected Gray code are well-known, e.g. [SALZ73].)

Radix Sorting

Due to the mixed-radix representation of the records the radix-sort algorithm (e.g. [REIN77]) naturally suggests itself. If we wished to sort the records in lexicographic order, i.e. according to L_1^m, then we would use the following algorithm:

form a list L from the set of records

for $j \leftarrow m$ **downto** 1 **do**

form empty lists $L_1, \cdots, L_{N_j}$

for each record X from list L **do**

append X to L_{x_j}

endfor

form a new L by concatenating $L_1, \cdots, L_{N_j}$

endfor

Briefly, its correctness is due to the fact that after i iterations L is sorted by its final i positions, i.e. with respect to L_{m-i+1}^m.

To produce the records in Gray-code order we note that $G_a^b = G_a^a \otimes G_{a+1}^b = [G_a^a(0)G_{a+1}^b, (G_a^a(1)G_{a+1}^b)^R, \cdots]$. Hence if we have sorted the records with respect to G_{a+1}^b and have divided them into N_a lists we find the odd lists need to be reversed. There are several ways to do this; perhaps the simplest is to replace the "append" statement above with:

if x_j even **then**

append X to L_{x_i}

else

prepend X to L_{x_i}

endfor.

The running time of these radix-sort algorithms are both $O(nm + \sum_{i=1}^{m} N_i)$. When the second term is small this approach should be quite fast.

References

[BITN76] J. R. Bitner, G. Ehrlich and E. M. Reingold, Efficient Generation of the Binary Reflected Gray Code and Its Applications, *Communications of the ACM*, **19**(9), 1976, pp. 517-521.

[ERNV79] J. Ernvall and O. Nevalainen, Compact Storage Schemes for Formatted Files by Spanning Trees, *BIT*, **19**, 1979, pp. 463-475.

[ERNV84] J. Ernvall, On the Construction of Spanning Paths by Gray-code in Compression of Files, *Technique et Science Informatiques*, **3**(6), 1984, pp. 411-414.

[GILB58] E. N. Gilbert, Gray Codes and Paths on the n-Cube, *Bell System Technical Journal*, **37**(9), 1958, pp. 815-826.

[KANG77] A. N. C. Kang, R. C. T. Lee, C. Chang and S. Chang, Storage Reduction Through Minimal Spanning Trees and Spanning Forests, *IEEE Transactions on Computers*, **C-26**(5), 1977, pp. 425-434.

[REIN77] E. M. Reingold, J. Nievergelt and N. Deo, *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall, 1977.

[ROBI81] J. Robinson and M. Cohn, Counting Sequences, *IEEE Transactions on Computers*, **C-30**(1), 1981, pp. 17-23.

[SALZ73] H. Salzer, Gray Code and the ± Sign Sequence, *Communications of the ACM*, **16**(3), 1973, pp. 180.