RJ 10192 (95066) August 17, 2000
Computer Science

Research Report

A SYSTEMATIC ENUMERATION OF CONTINGENCY TABLES

Byron E. Dom
IBM Research Division
650 Harry Rd.
San Jose, CA 95120

A SYSTEMATIC ENUMERATION OF CONTINGENCY TABLES

Byron E. Dom
IBM Research Division
650 Harry Rd.
San Jose, CA 95120

December 1, 2000

ABSTRACT: We derive an enumeration scheme for m-bin one-dimensional contingency tables that sum to n. This might be used for compression (encoding) or to generate indices into an array. The scheme corresponds to a *lexicographic* ordering of the tables.

Keywords: enumeration, enumerative coding

1. The Problem

We are concerned with one-dimensional contingency tables $\mathbf{t} = \{t_i | i = 1, 2, \ldots, m\}$, where the t_i are non-negative integers and sum to n: $\sum_i t_i = n$. Contingency tables correspond to counting discrete items. We seek an efficient encoding of these tables. We are motivated by the need to store a table T in memory where each entry corresponds to a different $\mathbf{t}$. We want to store these in the minimum amount of memory yet be able to access them efficiently. We will store the table as a linear array and we need to derive a scheme to rapidly compute the integer index into that array. Associating objects (contingency tables in this case) with all or some of the positive integers is the process of *enumeration.*

2. The Solution

Some Definitions:

- $\mathbf{t} = \{t_i | i = 1, 2, \ldots, m\}$ where $\sum_{i=1}^{m} t_i = n$.
- $\mathbf{t}^k = \{t_1, t_2, \ldots, t_k\}$ = the "prefix" table of length k; thus $\mathbf{t}^m \equiv \mathbf{t}$.
- $C^{(n,m)}$: the set of all (m, n) contingency tables $\mathbf{t}$.
- $\nu(n, m) \equiv |C^{(n,m)}|$ = the number of unique tables in $C^{(n,m)}$.
- $\iota(\mathbf{t})$ = index value associated (by our scheme) with table t.

We make use of the following lemma. See the appendix for a proof.

Theorem 2.1. The number $\nu(m, n)$ of m-bin one-dimensional contingency tables summing to n is:

$$\nu(n, m) = \binom{n+m-1}{m-1} \equiv \frac{(n+m-1)!}{(m-1)!\,n!}.$$

The enumeration scheme is straightforward. First, observe the following: the number of $\mathbf{t}$'s with $t_1 = 0$ is equal to $\nu(n, m-1)$, the number with $t_1 = 1$ is $\nu(n-1, m-1)$ and so on. In our scheme the indices 1 through $\nu(n, m-1)$ will correspond to tables with $t_1 = 0$. Indices $\nu(n, m-1) + 1$ through $\nu(n, m-1) + \nu(n-1, m-1)$ correspond to tables with $t_1 = 1$ and so on with the index value $\nu(n, m)$ going to the table with $t_1 = n$ and all other $t_i = 0$. We then apply the same idea recursively within each of those ranges of indices. Thus the first $\nu(n, m-2)$ indices go to tables for which both $t_1 = 0$ and $t_2 = 0$; the next: $\nu(n, m-2) + 1$ through $\nu(n, m-2) + \nu(n-1, m-2)$ go to tables for which $t_1 = 0$ and $t_2 = 1$, and so on. From this it should also be clear that the table with $t_m = n$ and all other $t_i = 0$ will be associated with index value 1.

Definitions:

$$\sigma(\mathbf{t}^k) \equiv \sum_{i=1}^{k} t_i$$

$$\eta(\mathbf{t}^k) \equiv \sum_{j=0}^{t_k-1} \nu(n - \sigma(t^{k-1}) - j, m - k) \qquad (1)$$

Theorem 2.2.

$$\iota(\mathbf{t}) = \sum_{i=1}^{m-1} \eta(\mathbf{t}^i) \qquad (2)$$

Proof: See paragraph above.
Note 1: This scheme can be shown to be a special case of the general enumerative encoding method described in [1]. This connection is demonstrated in Appendix B.
Note 2: The ordering produced by this scheme is *lexicographic*[1], which means the following. The contingency tables can be thought of as m-digit base-$(n+1)$ numbers. The first bin (t_1) of $\mathbf{t}$ is taken to me the most significant digit and the last (t_m) the least significant. The indices produced correspond to a numerically sorted list of the numbers corresponding to all $\nu(n,m)$ tables. See Appendix C.4 for an example.

3. Computational Issues

The number of ν values that must be summed is equal to $\sigma(t^{m-1}) \sim O(n)$.

4. Implementation and Experiments

To check the validity of (2) it was implemented in the APL function INDEX (see Appendix C.1). Complete sets of tables were generated in order using the APL functions MAKE_TABLES (see Appendix C.2) and BUILD_TABLE (see Appendix C.3). The index values from this ordering were then compared with those computed by INDEX. Sample results are presented in Appendix C.4.

Acknowledgment: Thanks to Corneliu Constantinescu for making me aware of the reference [1].

References

[1] Thomas M. Cover. Enumerative source encoding. *IEEE Transactions on Information Theory*, IT-19(1):73–77, January 1973.

Appendices

A. The Number of m-Bin Contingency Tables Summing to n

This is a well known result. It is derived here for pedagogical purposes.

Theorem A.1. The number $\nu(m,n)$ of m-bin one-dimensional contingency tables summing to n is:

$$\nu(n,m) \quad = \quad \binom{n+m-1}{m-1} \quad \equiv \quad \frac{(n+m-1)!}{(m-1)!\,n!}.$$

Proof: We will represent our tables abacus-style. Think of a linear (i.e. not a closed loop) string of n beads (each represented here by "$\cdot$") and $m-1$ movable partitions (represented by "$|$") that can be placed between the beads to delineate bins.

$$\cdot\cdot\cdot|\cdot|\cdot\ \ \ldots\ \ \cdot|\cdot$$

1. The number of possible arrangements (permutations) of these $n+m-1$ objects (beads plus partitions) is $(n+m-1)!$.
2. Each permutation corresponds to a table as follows. The count in the first bin of the table is the number of beads appearing to the left of the first (left-most) partition. The count in the 2nd bin is the number of beads between the first and second partitions and so on with the count in the m^{th} bin being the number of beads appearing to the right of the last (right-most) partition. Clearly the permutation-to-table mapping is many-to-one.
3. The n beads are indistinguishable from each other as are the $m-1$ partitions. Thus, to obtain the number of tables (unique sets of bin counts) we must simply divide our original number of permutations for the combined set (beads plus partitions) by the number of permutations of the partitions $(m-1)!$ times the number of permutations of the beads $n!$, yielding:

$$\nu(n,m) \quad = \quad \frac{(n+m-1)!}{(m-1)!\,n!}.$$

4. Q.E.D.

B. Connection with Previous work of Cover: "Enumerative Source Encoding"

Here we discuss the connection of our enumeration scheme with the general scheme of Cover[1]. Cover provides the following general enumeration scheme for length-m strings whose symbols are taken from an n-symbol alphabet.

$$\iota(\mathbf{t}) \quad = \quad \sum_{k=1}^{m} \sum_{j=1}^{t_k - 1} n_s(t_1, t_2, \ldots, t_{k-1}, j), \tag{3}$$

where the notation has been changed slightly to make it closer to ours. In the strings corresponding to (3) the alphabet consists of the integers 1 through n. The set S of strings to which this applies will, in general, be a subset of the complete set $\{1, 2, \ldots, n\}^n$. The constraints defining this subset are reflected in the counting function $n_s(t_1, t_2, \ldots, t_k) \equiv n_s(\mathbf{t}^k)$, which is defined as the number of strings in S with the prefix $\mathbf{t}^k \equiv \{t_1, t_2, \ldots, t_k\}$.

Applying this to our problem, in which $S = C^{(n,m)}$, we make the following identification:

$$n_s(\mathbf{t}^k) \quad = \quad \nu(n - \sigma(t^k), m - k).$$

Also, the "symbols" in our case are the possible bin values $\{0, 1, \ldots, n\}$. Thus we change the lower limit of the inner sum in (3) from 1 to 0. Finally, we note that the upper limit of the outer sum can be changed from m to $m - 1$ in our case because the $k = m$ term has a value of zero. Making these changes yields the following, which is our (2) with (1) substituted for η^k.

$$\iota(\mathbf{t}) \quad = \quad \sum_{k=1}^{m-1} \sum_{j=0}^{t_k - 1} \nu(n - \sigma(t^{k-1}) - j, m - k) \tag{4}$$

C. APL Program Listings

C.1. INDEX Function

```
[0]   I←INDEX T;M;N;K;I;SUM;X
[1]   ⍝ THIS FUNCTION RETURNS THE LEXICOGRAPHIC-ORDER
[2]   ⍝ INDEX I FOR THE 1-D CONTINGENCY TABLE T.
[3]   ⍝      AUTHOR: Byron Dom;   DATE: 08/09/2000
[4]    M←⍴T   ⍝ NO. BINS IN TABLE
[5]    N←+/T  ⍝ SUM OF TABLE
[6]    K←1    ⍝ BIN INDEX
[7]    I←0    ⍝ RESULT: ORDER INDEX OF T
[8]   LOOPK:
[9]    SUM←0++/(K-1)↑T    ⍝ SUM OF PREVIOUS BINS (HIGHER ORDER)
[10]   X←0                ⍝ VARIED OVER POSSIBLE VALUES OF BIN
[11]   →(T[K]=0)/SKIP     ⍝ IF THIS BIN=0, NOTHING TO ADD TO INDEX
[12]  LOOPX:
[13]   I←I+(M-(K+1))!N+(M-K)-(SUM+X+1) ⍝ ADD NO. TABLES WITH T[K]=X
[14]   →(T[K]>X←X+1)/LOOPX
[15]  SKIP:
[16]   →(M>K←K+1)/LOOPK
[17]   I←I+1
```

C.2. MAKE_TABLES Function

```
[0]   TTABLE←M MAKE_TABLES N;NROWS;L;T
[1]   ⍝
[2]   ⍝ THIS BUILDS A TWO-DIMENSIONAL TABLE WHOSE ROWS ARE ALL
[3]   ⍝ THE M-BIN ONE-DIMENSIONAL CONTINGENCY TABLES THAT SUM TO N.
[4]   ⍝ THESE TABLES (ROWS) ARE GENERATED IN LEXICOGRAPHIC ORDER.
[5]   ⍝
[6]   ⍝    AUTHOR: Byron Dom;    DATE: 08/09/2000
[7]   ⍝
[8]    TTABLE←(0,M)⍴⍳0   ⍝ FINAL RESULT (2-D TABLE)
[9]    T←⍳0              ⍝ USED TO BUILD 1-D CONT. TABLES BY BUILD_TABLE
[10]   L←0               ⍝ INDEX FOR RECURSION LEVEL → CURRENT BIN IN T.
[11]   BUILD_TABLE    ⍝ RECURSIVE FUNCTION THAT ACTUALLY DOES IT
[12]   NROWS←1↑⍴TTABLE
[13]   TTABLE←((NROWS,1)⍴⍳NROWS),TTABLE ⍝ 1ST COL. IS INDEX; THE REST IS TABLE.
```

C.3. BUILD_TABLE Function

```
[0]   BUILD_TABLE;I;SUM
[1]   ⍝
[2]   ⍝ RECURSIVE FUNCTION CALLED BY THE FUNCTION 'MAKE_TABLES'
[3]   ⍝ TO BUILD A 2-D ARRAY WHOSE ROWS ARE ALL M-BIN 1-D CONTINGENCY
[4]   ⍝ TABLES THAT SUM TO N. THEY ARE GENERATED IN LEXICOGRAPHIC ORDER.
[5]   ⍝     AUTHOR: Byron Dom;    DATE: 08/09/2000
[6]   ⍝
[7]    L←L+1     ⍝ L IS RECURSION LEVEL; THE BIN BEING VARIED BY THIS INVOCATION
[8]    SUM←+/T  ⍝ T IS 1-D CONT. TABLE; SUM IS SUM OF BINS 1 THRU L-1
[9]    →(M=L)/BOTTOM  ⍝ IS THIS THE LAST BIN?
[10]  ⍝
[11]  ⍝ THE FOLLOWING LOOPS THROUGH ALL POSSIBLE VALUES OF
[12]  ⍝ THE Lth (CURRENT) TABLE ENTRY AND RECURSIVELY BUILDS
[13]  ⍝ THE REST OF THE TABLE.
[14]  ⍝
[15]   I←0
[16]   T←L↑T
[17]  LOOP:
[18]   T[L]←I
[19]   BUILD_TABLE           ⍝ BUILDS THE REST OF T, BINS L+1 THRU M.
[20]   L←L-1
[21]   T←L↑T
[22]   →((N-SUM)≥I←I+1)/LOOP
[23]   →0
[24]  ⍝
[25]  ⍝ THE FOLLOWING EXECUTES AT THE 'BOTTOM' OF THE RECURSION
[26]  ⍝ (THE LAST BIN), WHICH CORRESPONDS TO ONE COMPLETE 1-D TABLE
[27]  ⍝
[28]  BOTTOM:
[29]   T←T,N-SUM
[30]   TTABLE←TTABLE,[1]T
[31]   →0
```

C.4. Some Test Results

```
      3 MAKE_TABLES 5
 1 0 0 5
 2 0 1 4
 3 0 2 3
 4 0 3 2
 5 0 4 1
 6 0 5 0
 7 1 0 4
 8 1 1 3
 9 1 2 2
10 1 3 1
11 1 4 0
12 2 0 3
13 2 1 2
14 2 2 1
15 2 3 0
16 3 0 2
17 3 1 1
18 3 2 0
19 4 0 1
20 4 1 0
21 5 0 0
      INDEX 1 0 4
7
      INDEX 2 3 0
15
      INDEX 4 1 0
20
      INDEX ⍳ 10
4816547511
      +/⍳ 10
55
```