ISE-TR-83-40

ADDRESS CALCULATION ALGORITHMS FOR ORDERED SETS OF COMBINATIONS

by

Seiichi Nishihara

Katsuo Ikeda

July 15, 1983

INSTITUTE
OF
INFORMATION SCIENCES AND ELECTRONICS

UNIVERSITY OF TSUKUBA

Address Calculation Algorithms for Ordered Sets of Combinations

Seiichi NISHIHARA and Katsuo IKEDA

Institute of Information Sciences and Electronics

University of Tsukuba

Sakura-mura, Niihari-gun

Ibaraki 305, Japan

Keywords: Combination, address calculation, ordered set

1. Introduction

Our problem is to find algorithms for calculating addresses of given combinations arranged in lexicographic order. Consider the set of all combinations of M integers, $\{1, \ldots, M\}$, taken N at a time. The size of the set is $\binom{M}{N}$. Let $(u_1, \ldots, u_N)$ and $(v_1, \ldots, v_N)$ be two different N-combinations whose components are arranged in ascending order, i.e. $u_i \le u_{i+1}$, $v_i \le v_{i+1}$ for $1 \le i < N$. Then there is some k such that $u_k \neq v_k$, and $u_i = v_i$ for $i=1, \ldots, k-1$. Let us call such a k the first different position of components. In other words, k is the minimum subscript that suffices $u_k \neq v_k$. An ordering relation is defined as $(u_1, \ldots, u_N) \prec (v_1, \ldots, v_N)$ when $u_k < v_k$, where k is the first different position of components, and vice versa. This relation, called lexicographic ordering, is a total ordering. Now arrange all the N-combinations in lexicographic order, and assign sequential numbers, or addresses, from 0 to $\binom{M}{N} - 1$.

Table 1 shows an example of an ordered table of combinations, where M=8, N=4. The address ranges from 0 to 69 ($= \binom{8}{4} - 1$). Generally, the first combination located at the address 0 is $(1, 2, \ldots, N)$. The last one located at the address $\binom{M}{N} - 1$ is $(M-N+1, \ldots, M-1, M)$.

2. The algorithms

In this section algorithms for calculating the address of a given combination are presented. Firstly we consider the ordered sets of combinations of M integers taken N at a time. Next the algorithm is extended to the case that permits repetitions.

2.1 Addressing of combinations without repetitions

Table 1. Ordered table of combinations without repetitions, where M=8, N=4.

addr					addr				
0	1	2	3	4	35	2	3	4	5
1	1	2	3	5	36	2	3	4	6
2	1	2	3	6	37	2	3	4	7
3	1	2	3	7	38	2	3	4	8
4	1	2	3	8	39	2	3	5	6
5	1	2	4	5	40	2	3	5	7
6	1	2	4	6	41	2	3	5	8
7	1	2	4	7	42	2	3	6	7
8	1	2	4	8	43	2	3	6	8
9	1	2	5	6	44	2	3	7	8
10	1	2	5	7	45	2	4	5	6
11	1	2	5	8	46	2	4	5	7
12	1	2	6	7	47	2	4	5	8
13	1	2	6	8	48	2	4	6	7
14	1	2	7	8	49	2	4	6	8
15	1	3	4	5	50	2	4	7	8
16	1	3	4	6	51	2	5	6	7
17	1	3	4	7	52	2	5	6	8
18	1	3	4	8	53	2	5	7	8
19	1	3	5	6	54	2	6	7	8
20	1	3	5	7	55	3	4	5	6
21	1	3	5	8	56	3	4	5	7
22	1	3	6	7	57	3	4	5	8
23	1	3	6	8	58	3	4	6	7
24	1	3	7	8	59	3	4	6	8
25	1	4	5	6	60	3	4	7	8
26	1	4	5	7	61	3	5	6	7
27	1	4	5	8	62	3	5	6	8
28	1	4	6	7	63	3	5	7	8
29	1	4	6	8	64	3	6	7	8
30	1	4	7	8	65	4	5	6	7
31	1	5	6	7	66	4	5	6	8
32	1	5	6	8	67	4	5	7	8
33	1	5	7	8	68	4	6	7	8
34	1	6	7	8	69	5	6	7	8

Assume an N-combination $U=(u_1, \ldots, u_N)$ is given, where $u_i<u_{i+1}$ for $1 \leq i<N$. Let $S_i(u_1, \ldots, u_i)$ denote the set of all combinations whose first i components are equal to $u_1, \ldots, u_i$. All the combinations in $S_i(u_1, \ldots, u_i)$ are placed consecutively in an ordered table. Further, notice that the size of $S_i(u_1, \ldots, u_i)$ is equal to the number of (N-i)-combinations taken out of $M-u_i$ elements, $\{u_i+1, u_i+2, \ldots, M\}$, i.e.

$$\left|S_i(u_1, \ldots, u_i)\right| = \binom{M-u_i}{N-i}, \tag{1}$$

where $|X|$ denotes the size of set X. Note that the first one, i.e. the one having the lowest address, of those combinations is $(u_1, \ldots, u_i, u_i+1, \ldots, u_i+N-i)$. Let A_i denote the address of combination $(u_1, \ldots, u_i, u_i+1, \ldots, u_i+N-i)$ for $i=0,1, \ldots, N-1$. Then, the address A_{i+1} of combination $(u_1, \ldots, u_i, u_{i+1}, u_{i+1}+1, \ldots, u_{i+1}+N-i-1)$, which is the first combination of $S_{i+1}(u_1, \ldots, u_{i+1})$ in lexicographic order, is given as follows:

$$\begin{aligned} A_{i+1} &= A_i + \left|S_{i+1}(u_1, \ldots, u_i, u_i+1)\right| + \left|S_{i+1}(u_1, \ldots, u_i, u_i+2)\right| + \ldots \\ &\quad + \left|S_{i+1}(u_1, \ldots, u_i, u_{i+1}-1)\right| \\ &= A_i + \sum_{j=1}^{u_{i+1}-u_i-1} \left|S_{i+1}(u_1, \ldots, u_i, u_i+j)\right| . \end{aligned}$$

Thus, by using (1),

$$A_{i+1} = A_i + \sum_{j=1}^{u_{i+1}-u_i-1} \binom{M-(u_i+j)}{N-(i+1)}$$

for $i=0, \ldots, N-1$, where $A_0=0$ and $u_0=0$. In particular, if $u_{i+1}=u_i+1$, then $A_{i+1}=A_i$. The address of a given combination $U=(u_1, \ldots, u_N)$, denoted by $addr_{M,N}(U)$, is equal to A_N, thus

$$addr_{M,N}(U)= \sum_{i=1}^{N} \sum_{j=1}^{d_i} \binom{M-u_{i-1}-j}{N-i}, \qquad (2)$$

where $d_i=u_i-u_{i-1}-1$.

Example 1.

Let M=8, N=4, for which the ordered table is shown as Table 1. The address of the given combination U=(3,5,6,8) is calculated by using formula (2) as follows.

$$addr_{8,4}((3,5,6,8))= \sum_{j=1}^{2} \binom{8-j}{3} + \sum_{j=1}^{1} \binom{8-3-j}{2} + \sum_{j=1}^{0} \binom{8-5-j}{1} + \sum_{j=1}^{1} \binom{8-6-j}{0}$$

$$= \binom{7}{3} + \binom{6}{3} + \binom{4}{2} + \binom{1}{0}$$

$$= 62.$$

2.2 Addressing of combinations with repetitions

Next the algorithm given in Section 2.1 is extended to the case of combinations with repetitions permitted. Table 2 shows an example of an ordered table where M=6, N=3. In general, the first combination located at the address 0 is $(1, \ldots, 1)$, while the last one is $(M, \ldots, M)$. The size of the ordered table is equal to the number of N-combinations of M elements with repetitions permitted, i.e. $\binom{M+N-1}{N}$,[1]. Let $U=(u_1, \ldots, u_N)$ is a given combination, where $u_i \leq u_{i+1}$ for $1 \leq i < N$. Let $S_i'(u_1, \ldots, u_i)$ denote the set of all combinations with repetitions permitted whose first i components are equal to $u_1, \ldots, u_i$. Then the size of $S_i'(u_1, \ldots, u_i)$ is given as the number of (N-i)-combinations with repetitions permitted taken out of $M-u_i+1$ elements, $\{u_i, u_i+1, \ldots, M\}$, i.e.

$$\left|S_i'(u_1, \ldots, u_i)\right| = \binom{M-N-u_i-i}{N-i}. \qquad (1')$$

Let A_i' denote the address of combination $(u_1, \ldots, u_i, u_i, \ldots, u_i)$, which is the first combination of $S_i'(u_1, \ldots, u_i)$ in lexicographic order. Then the address A_{i+1}' of combination $(u_1, \ldots, u_i, u_{i+1}, u_{i+1}, \ldots, u_{i+1})$, which is the first combination of $S_{i+1}'(u_1, \ldots, u_{i+1})$ is given in a similar manner to the case without repetition as follows:

$$A_{i+1}' = A_i' + \sum_{j=1}^{u_{i+1}-u_i} \left|S_{i+1}(u_1, \ldots, u_i \; u_i+j-1)\right|$$

$$= A_i' + \sum_{j=1}^{u_{i+1}-u_i} \binom{M+N-(u_i+j-1)-(i+1)}{N-(i+1)},$$

for $i=0, \ldots, N-1$, where $A_0'=0$, $u_0=1$. In particular, if $u_{i+1}=u_i$, then $A_{i+1}'=A_i'$. Therefore, the address of a given combination with repetitions permitted, $U=(u_1, \ldots, u_N)$, is given as

$$\mathrm{addr}_{M,N}'(U) = \sum_{i=1}^{N} \sum_{j=1}^{d_i'} \binom{M+N-u_{i-1}-j-i+1}{N-i}, \qquad (2')$$

where $d_i'=u_i-u_{i-1}$.

Example 2.

Let M=6 and N=3, for which the ordered table with repetition permitted is shown as Table 2. The address of a given combination U=(3,5,5) is calculated by formula (2′) as

Table 2. Ordered table of combinations with repetitions permitted, where M=6, N=3.

addr'				addr'			
0	1	1	1	28	2	3	5
1	1	1	2	29	2	3	6
2	1	1	3	30	2	4	4
3	1	1	4	31	2	4	5
4	1	1	5	32	2	4	6
5	1	1	6	33	2	5	5
6	1	2	2	34	2	5	6
7	1	2	3	35	2	6	6
8	1	2	4	36	3	3	3
9	1	2	5	37	3	3	4
10	1	2	6	38	3	3	5
11	1	3	3	39	3	3	6
12	1	3	4	40	3	4	4
13	1	3	5	41	3	4	5
14	1	3	6	42	3	4	6
15	1	4	4	43	3	5	5
16	1	4	5	44	3	5	6
17	1	4	6	45	3	6	6
18	1	5	5	46	4	4	4
19	1	5	6	47	4	4	5
20	1	6	6	48	4	4	6
21	2	2	2	49	4	5	5
22	2	2	3	50	4	5	6
23	2	2	4	51	4	6	6
24	2	2	5	52	5	5	5
25	2	2	6	53	5	5	6
26	2	3	3	54	5	6	6
27	2	3	4	55	6	6	6

follows.

$$\mathrm{addr}'_{6,3}((3,5,5)) = \sum_{j=1}^{2}\binom{6+3-1-j-1+1}{3-1} + \sum_{j=1}^{2}\binom{6+3-3-j-2+1}{3-2} + \sum_{j=1}^{0}\binom{6+3-5-j-3+1}{3-3} = 43 .$$

2.3 Algorithms expressed in Pascal programs

The algorithms given above for calculating addresses of given combinations are summarized more explicitly in the form of Pascal programs;

```
function addr:integer;
var i,j,a:integer;
begin
    u[0]:=0;
    a:=0;
    for i:=1 to N do
        for j:=1 to u[i]-u[i-1]-1 do
            a:=a+Comb(M-u[i-1]-j,N-i);
    addr:=a
end;
```

for formula (2), and

```
function addr':integer;
var i,j,a:integer;
```

```
begin
    u[0]:=1;
    a:=0;
    for i:=1 to N do
        for j:=1 to u[i]-u[i-1] do
            a:=a+Comb(M+N-u[i-1]-j-i+1,N-i);
    addr':=a
end;
```

for formula (2'), where M, N and U=(u[1], u[2], . . . ,u[N]) are assumed to be declared and given values outside each procedure. The procedure Comb(m,n) computes the binomial coefficient, or $\binom{m}{n}$, whose implementation technique is briefly discussed in the Appendix.

3. Conclusion

Algorithms that calculate the addresses of given N-combinations of M integers arranged in lexicographic order are discussed. Two concrete procedures for combinations with and without repetitions are presented. Our techniques may be applicable for searching data composed of more than one attribute-value pair.

Acknowledgment

The authors thank Professor J. W. Higgins of the Institute of Information Sciences and Electronics of the University of Tsukuba for his valuable suggestions.

References

[1] D. E. Knuth, The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 2nd ed. (Addison-Wesley, Reading, MA, 1973).

[2] T. Iriyama's program (introduced by S. Ohkoma in Japanese), **bit**, 15, 4 (1983), 409-413.

Appendix

The computations of formula (2) and (2') can be performed by recursive procedures. Let us define

$$s(a,b,c)= \sum_{i=1}^{b} \binom{a-i}{c} ,$$

for a,b,c>0. Then,

$$s(a,b,c)=s(a-1,b,c)+s(a-1,b,c-1).$$

Especially,

$$s(a,b,c)=0 \qquad \text{for } b=0 \text{ or } a \leq c,$$

and

$$s(a,b,c)=b \qquad \text{for } a \geq b \text{ and } c=0.$$

Formula (2), for example, is rewritten in terms of function s as

$$addr_{M,N}(U)= \sum_{i=1}^{N} s(M-u_{i-1},d_i,N-i).$$

Therefore, formula (2) can be expressed in recursive form. However, the cost to execute these recursive procedures becomes very high when M and N increase.

Here we should refer to a direct execution algorithm for the binomial coefficient $\binom{m}{n}$

(=Comb(m,n) in our procedures) which was given by T. Iriyama[2]. We can formulate $\binom{m}{n}$ ($m \geq n > 0$) as $\binom{a+b}{b}$, where $b=n$, $a=m-n \geq 0$, without loss of generality. By definition,

$$\binom{a+b}{b} = \frac{(a+1)\cdot(a+2)\cdot \ldots \cdot(a+b)}{1 \cdot 2 \cdot \ldots \cdot b}$$

Let us assume there are two arrays as

A[i]=a+i,

B[i]=i, for $1 \leq i \leq b$.

Notice that a multiple of B[i] is always found in A[i-mod(a,i)], for $1 \leq i \leq N$, where mod(a,i) means the remainder of a divided by i. Further, the values of B[i+i·k] and A[i-mod(a,i)+i·k] for k=0,1, . . . , $\lfloor (b-i)/i \rfloor$ are all multiples of B[i]. ($\lfloor x \rfloor$ means the floor of x, [1].) Now, for i=2 to b, repeat the execution of all possible division operations for elements in A and B by B[i]. Then all elements in B become equal to 1. Thus the binomial coefficient is finally derived by just multiplying all the A[i]'s. For example,

$$\binom{7}{4} = \frac{\overset{2}{\cancel{4}}\cdot 5\cdot \overset{3}{\cancel{6}}\cdot 7}{1\cdot \underset{1}{\cancel{2}}\cdot 3\cdot \underset{2}{\cancel{4}}} = \frac{2\cdot 5\cdot \overset{1}{\cancel{3}}\cdot 7}{1\cdot 1\cdot \underset{1}{\cancel{3}}\cdot 2}$$

$$= \frac{\overset{1}{\cancel{2}}\cdot 5\cdot 1\cdot 7}{1\cdot 1\cdot 1\cdot \underset{1}{\cancel{2}}} = \frac{1\cdot 5\cdot 1\cdot 7}{1\cdot 1\cdot 1\cdot 1}$$

$$= 35 .$$

Thus formulas (2) and (2') can directly be computed by using this algorithm.

A concrete program for calculating the binomial coefficient, $\binom{m}{n}$, is given as follows.

```
function Comb(m,n:integer):integer;
    var A,B:array[1..50] of integer;
        i,j,k,p,w:integer;
    begin
        if (n>m-n) then n:=m-n;
        for i:=1 to n do
            begin
                A[i]:=m-n+i;
                B[i]:=i
            end;
        for k:=2 to n do
            begin
                p:=B[k];
                if (p>1) then
                    begin
                        i:=k;
                        j:=(m-n) mod i;
                        repeat A[i-j]:=A[i-j] div p;
                               B[i]:=B[i] div p;
                               i:=i+k
                        until i>n
                    end
            end;
        w:=1;
        for i:=1 to n do
            if A[i]>1 then w:=w*A[i];
        Comb:=w
    end;
```

```
program addr(input,output);
label 1;
var M,N,sw,i: integer;
    u: array[0..50] of integer;
(**)

(**addr**)
    function addr:integer;
    var i,j,a:integer;
    begin
        u[0]:=0;
        a:=0;
        for i:=1 to N do
            for j:=1 to u[i]-u[i-1]-1 do
                a:=a+Comb(M-u[i-1]-j,N-i);
        addr:=a
    end;
(**addrp**)
    function addrp:integer;
    var i,j,a:integer;
    begin
        u[0]:=1;
        a:=0;
        for i:=1 to N do
            for j:=1 to u[i]-u[i-1] do
                a:=a+Comb(M+N-u[i-1]-j-i+1,N-i);
        addrp:=a
    end;
(****)
begin
    1: read(M,N,sw);
    if M<>0 then
        begin
            for i:=1 to N do read(u[i]);
            if (sw=0) then writeln(M,N,sw,addr)
                      else writeln(M,N,sw,addrp);
            goto 1
        end
end.
```

INSTITUTE OF INFORMATION SCIENCES AND ELECTRONICS
UNIVERSITY OF TSUKUBA
SAKURA-MURA, NIIHARI-GUN, IBARAKI 305 JAPAN

REPORT DOCUMENTATION PAGE	REPORT NUMBER ISE-TR-83-40

TITLE

Address Calculation Algorithms for Ordered Sets of Combinations

AUTHOR(S)

Seiichi Nishihara
Katsuo Ikeda
Inst. of Information Sciences and Electronics,
University of Tsukuba

REPORT DATE	NUMBER OF PAGES
July 15, 1983	13
MAIN CATEGORY	**CR CATEGORIES**
Data Storage Representation	E.2, E.1, F.2.2, G.2.1

KEY WORDS

combination, address calculation, table, retrieval, ordered set

ABSTRACT

Algorithms that calculate the addresses of given N-combinations of M integers, $\{1, 2, \ldots, M\}$, arranged in lexicographic order are discussed. Two concrete procedures for combinations with and without repetitions are presented. Our techniques may be applicable for searching data composed of more than one attribute-pair.

SUPPLEMENTARY NOTES