Annals of Discrete Mathematics 2 (1978) 135–148.

A UNIFIED SETTING FOR SELECTION ALGORITHMS (II)*

Herbert S. WILF
Department of Mathematics, University of Pennsylvania, PA 19174, U.S.A.

In [3] we studied a general framework in which algorithms for sequencing, ranking and random selection of combinatorial objects can be carried out in a unified manner. Roughly speaking, the members of a combinatorial family which has a recursive structure can be encoded as paths in a certain graph G, which depends only on the family. We can then perform the desired selection algorithms on the paths issuing from a fixed vertex (the "order"), and either use the output information in coded form, if possible in a particular application, or else decode to obtain the familiar representation of the object.

In [3] we applied the method to several familiar combinatorial families, such as sets, partitions, permutations, subspaces, tableaux, etc., and further applications have been uncovered. Klingsberg, for example, has found that labelled trees on n vertices with k terminal vertices fit into these structures, and has $O(n)$ algorithms, including decoding time, for all of the above operations.

Here we further develop the ideas of [3] by first displaying quite explicitly the initial portions of the relevant graphs for three families — subsets, partitions of sets with given number of classes, and permutations of n letters with k cycles (see Appendices 1, 2, 3) along with lexicographically arranged lists, for these three families, of the objects of order (5, 3), giving rank, codeword, and familiar form, in each case.

Next we discuss two families which are covered by the general theory, the first, compositions of n into k parts, being new here, the second, Young tableaux, having been mentioned in [3], but treated in more detail here. Third we consider the interesting question of the compactness of the coding scheme, *i.e.*, the economy of the representation, which leads to several difficult problems. Finally, we give quite explicit general sequencing algorithms.

1. Structure of the graphs

For an illustration of the method, we shall trace one of the paths in Fig. A2, Appendix 2, namely the path 23000, or in detail:

* Research supported by the National Science Foundation. Prepared as a talk for the Conference on Algorithms in Combinatorics, Qualicum Beach, B.C., Canada, May, 1976.

$$(5,3)\xrightarrow{2}(4,3)\xrightarrow{3}(3,2)\xrightarrow{0}(2,2)\xrightarrow{0}(1,1)\xrightarrow{0}(0,0).$$

Now Fig. A2 is the graph for partitions of n-sets into k classes, and each path from (n,k) to $(0,0)$ represents such a partition. Our chosen path, therefore, represents a partition of $\{1,2,3,4,5\}$ into 3 classes. We discover which partition it is by following the path backwards from $(0,0)$.

The last edge is diagonal, and it carries the road sign "create a new class, and enter 1 into it". Our partition is, so far, (1). We traverse the next edge backwards, to $(2,2)$. It is diagonal, and its sign reads "create a new class, and enter 2 into it". Our partition is now (1) (2). Next we encounter a horizontal edge 0, and so its command to the traveller is "enter 3 into the 0^{th} class", so we have (13) (2). Next the diagonal edge 3 orders us to "create a new class and enter 4 into it," and we have (13) (2) (4), while finally horizontal edge 2 instructs us to "enter 5 into class 2" (the classes are now numbered $0,1,2$) and so we have our partition

(13) (2) (45),

Now each edge carries *two* permanent road signs. One of them is, as we have seen, a command as to the disposition of the x-coordinate of its initial vertex in the output object. Any path which traverses the edge must execute that instruction.

The second permanent road sign on each edge is the ranking function $f(e)$ (see [3]) which is shown in a box near each edge of Fig. A2. In order to discover the rank of any particular walk we need only add up the ranking functions on each edge of the walk. Thus, on the walk described above, as we walk backwards we encounter successively, the values

$$0,0,0,3,12$$

of this function, and since the sum is 15, we have just taken the walk number 15 in the lexicographically ordered list of 25 partitions (numbered 0 to 24) of a 5-set into 3 classes. We have now completely accounted for one line of Table A2, Appendix 2.

2. Compositions of n into k parts

It is, of course, well known that there are

$$b(n,k)=\binom{n+k-1}{k-1} \tag{1}$$

compositions of n into k parts (see e.g. [2]). For our present purposes, observe that there are $b(n,k-1)$ compositions whose first part is 0, and $b(n-1,k)$ whose first part is positive, since each of the latter is obtained by adding 1 to the first part of a composition of $n-1$ into k parts. Hence we have the recurrence

$$b(n,k)=b(n,k-1)+b(n-1,k), \tag{2}$$

which is also evident from (1), and its combinatorial meaning.

The form of (2) is unusual in that the two edges out of vertex (n, k) are horizontal and vertical, rather than horizontal and diagonal as has mostly been the case. The graph G has for vertices all lattice points (n, k) for which $n \geq 0$, $k \geq 1$, and the origin $(0,0)$, which is the terminal vertex. From each vertex (n, k) there go out edges to $(n-1, k)$, if $n \geq 1$, and to $(n, k-1)$, if $k \geq 2$. Finally, one edge goes from $(0, 1)$ to $(0, 0)$. All edges are numbered 0 or 1. If $n \geq 1$ and $k \geq 2$, the westbound edge out of (n, k) is labelled 0 and the southbound edge is 1. From other vertices there is just one edge, labelled 0. The functions φ, ψ are given by

$$\text{(a)} \quad \varphi(\mu, \nu) = \begin{cases} 1 & \text{if } \mu \geq 1 \quad \text{and} \quad \nu \geq 1 \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{(b)} \quad \psi(\mu, \nu) = \begin{cases} 1 & \text{if } \mu \geq 0 \quad \text{and} \quad \nu \geq 2 \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The constructive "road sign" on a westbound edge $(n, k) \rightarrow (n-1, k)$ is "add 1 to the first part of your composition", while on a southbound edge (n, k) to $(n, k-1)$ it reads "adjoin a new first part, equal to 0, to your composition". The other road sign, the ranking function, is calculated as described in [3]. The result is that $f(e)$, on an edge from (n, k), is given by

$$f(e) = \begin{cases} \binom{n+k-2}{k-1} & \text{if the number of the edge is 1} \\ 0 & \text{otherwise.} \end{cases}$$

3. Young tableaux

In [3] we described a conceptual algorithm for sequencing Young tableaux; here we discuss its implementation. In order to obtain the successor of a tableau T we must

(a) locate the smallest letter j, in T, which does not occupy the last position (i.e., last column of the last row) of the sub-tableau T_j formed by the letters $1, 2, \ldots, j$.

(b) move the letter j to the next lower corner position in T_j.

(c) replace all other entries in the shape T_j by entering the letters $1, 2, \ldots, j-1$ consecutively from top to bottom down the columns beginning with the first column.

To carry out this procedure it is helpful to use the *Yamanouchi vector* $y(1), \ldots, y(n)$, which is associated with a tableau T by letting $y(i)$ denote the row of T which contains i. Then T is uniquely recoverable from y, and step (a) of our algorithm becomes very easy: locate the first j such that $y(j) < y(j-1)$.

It follows that *the sequencing of Young tableaux which is given by our general*

machinery is identical with lexicographic ordering of the sequence of reversed Yamanouchi vectors corresponding to the given shape.

For the shape $(3,2,1)$, for example, the sequence thus generated of sixteen tableaux and their vectors is shown below,

1 4 6	1 3 6	1 2 6	1 3 6	1 2 6	1 4 5
2 5	2 5	3 5	2 4	3 4	2 6
3	4	4	5	5	3
(123121)	(121321)	(112321)	(121231)	(112231)	(123112)

1 3 5	1 2 5	1 3 4	1 2 4	1 2 3
2 6	3 6	2 6	3 6	4 6
4	4	5	5	5
(121312)	(112312)	(121132)	(112132)	(111232)

1 3 5	1 2 5	1 3 4	1 2 4	1 2 3
2 4	3 4	2 5	3 5	4 5
6	6	6	6	6
(121213)	(112213)	(121123)	(112123)	(111223)

The complete formal algorithm follows.

Next Young tableau of shape $(\lambda_1, \lambda_2, \ldots, \lambda_k)$. (Enter with a shape λ and the Yamanouchi vector y of a tableau. Exit with the vector y for the successor tableau.)

(A) [Enter here with y-vector to get next tableau] $\lambda(1) \leftarrow 1$; $\lambda(i) \leftarrow 0$ $(i = 2, k)$ [The array λ will now be set to the shape of the subtableau of the letters $\{1, \ldots, j\}$, where j is the point of decrease of y].

For $j = 2, n$ **do**:

$\lambda(y_j) \leftarrow \lambda(y_j) + 1$; If $y_j < y_{j-1}$ to (B)

End

Final exit.

(B) [Find new row i for letter j] $t \leftarrow \lambda(1 + y_j)$; $i \leftarrow k$;

While $\lambda(i) \neq t$ **do**:

$i \leftarrow i - 1$

End

[Move j to row i] $y_j \leftarrow i$; $\lambda(i) \leftarrow \lambda(i) - 1$

(C) [Enter here with shape λ to get first tableau] $t \leftarrow \lambda(1)$; $\bar{\lambda}_i \leftarrow 0$ $(i = 1, t)$; [Form $\bar{\lambda}$, the conjugate partition to λ]

For $r = 1, k$ **do**:

If $\lambda(r) = 0$, to (D)

For $s = 1$, $\lambda(r)$ **do**:

$\bar{\lambda}_s \leftarrow 1 + \bar{\lambda}_s$

End

End

(D) [Fill the entries of y vector corresponding to conjugate partition $\bar{\lambda}$]
$l \leftarrow 1$
For $r = 1, t$ **do**:
For $i = 1, \bar{\lambda}_r$ **do**:
$y_l \leftarrow i; \quad l \leftarrow l + 1$
End
End
Exit □

4. Compactness of coding

Among all ways of assigning distinct positive integers to members of a combinatorial family which contains N members, the most compact code simply assigns j to the j^{th} object ($j = 1, N$), and thereby uses a total of

$$\sum_{j=1}^{N} \lceil \log_2 j \rceil \sim (\log_2 e) N \log N \tag{4}$$

bits of storage.

Any scheme, for example, of coding the partitions of n must use at least

$$(\log_2 e) p(n) \log p(n) \tag{5}$$

bits, in all, and so in any such coding scheme there must be an average of

$$\begin{aligned} (\log_2 e) \log p(n) &\sim (\log_2 e) \pi \sqrt{\tfrac{2}{3}}\, n^{\frac{1}{2}} \\ &= 3 \cdot 700656 \sqrt{n} \end{aligned} \tag{6}$$

bits per partition.

In [3] we noted that the natural coding of partitions was via a string of 0's and 1's, where each 0 means "replicate the largest part" and each 1 means "add 1 to the largest part". The partition

$$23 = 6 + 4 + 3 + 3 + 3 + 2 + 1 + 1 \tag{7}$$

is coded by the string (read left to right)

1001010001011.

Pictorially, this coding amounts to walking along the profile of the Ferrers

diagram. For instance, with the partition (7)

(9)

of 23, each "1" bit is a horizontal step in tracing the profile and each 0 is a vertical step of one unit.

To code a general partition

$$n = r_1 + r_2 + \cdots + r_k \qquad (r_1 \geq r_2 \geq \cdots \geq r_k) \tag{10}$$

by this scheme requires $r_1 + k - 1$ bits. The average number of bits is therefore $2u(n) - 1$, where $u(n)$ is the average largest part of a partition of n. This function has been studied by Erdös and Lehner [1] with the result that

$$u(n) \sim cn^{\frac{1}{2}} \log n. \tag{11}$$

Our coding system is therefore (compare (6), (11)) slightly uneconomical of space. The reason for the wasted space is clear from comparison of (8) and (9). On a horizontal edge of the Ferrers diagram of length 5, for instance, we would write 11111 instead of the more compact binary number 101. It would be interesting to know how closely the average number of bits in the coding which uses place value compares with the minimum (6). This coding would consist in giving

$$d_1, \mu_1, d_2 - d_1, \mu_2, d_3 - d_2, \mu_3, \ldots, d_k - d_{k-1},$$

where $d_1 < d_2 < \cdots < d_k$ are the *distinct* parts of the partition, and the μ_i are their multiplicities.

Another family studied in [3], the vector subspaces, is easier to analyze. Indeed, the usual way to "give" a subspace is via a $k \times n$ basis matrix of field elements, which occupies about $kn \log_2 q$ bits. In our coding system, we specify a codeword of n edges, each edge bearing an index $\leq q^k$, for a total of $kn \log_2 q$ bits again, so that in this case the coding compares favorably with the conventional presentation.

5. The labor in sequencing

In passing from an object ω to its immediate successor $S(\omega)$ in the lexicographic list we back up along the walk ω until we reach an edge which is not the last one from its initial vertex. Suppose this edge is the j^{th} edge from the end of the walk. We study here the statistics of $j = j(\omega)$, since it is a measure of the labor involved in a step.

For each vertex v in the graph G let

$$J(v) = \sum_{\omega} j(\omega), \tag{12}$$

where the sum is over objects of order v. Suppose that all objects of order v have the same length n. Then, the lexicographic sequencing is such that the recurrence

$$J(v) = \sum_{v'} g_{vv'} J(v') + \rho_0(v) \qquad (J(\tau) = 0) \tag{13}$$

holds. Here, as in [3], $g_{vv'}$ is the number of edges from v to v' and $\rho_0(v)$ is the outvalence of v. To see why (15) holds, let $\Omega(v)$ denote the lexicographic sequence of all objects of order v. Then, of course,

$$\Omega(v) = 0 \otimes \Omega(\text{fin}(0)), \qquad 1 \otimes \Omega(\text{fin}(1)), \ldots, (\rho_0 - 1) \otimes \Omega(\text{fin}(\rho_0 - 1))$$

and (13) follows. Equation (13) should be compared with (1) of [3], the recurrence for $b(v)$.

We illustrate with a few examples. In the family of k-subsets of an n-set (13) becomes

$$J(n, k) = J(n-1, k) + J(n-1, k-1) + 2 \qquad (1 \leqslant k < n)$$

and if we put

$$J_n(x) = \sum_{k=1}^{n-1} J(n, k) x^k,$$

then we find

$$J_n(x) = (1+x) J_{n-1}(x) + 2(x + x^2 + \cdots + x^{n-1}),$$

with solution

$$J_n(x) = 2 \sum_{k=1}^{n} (1+x)^{n-k} \{x + x^2 + \cdots + x^{k-1}\}.$$

The average values of j, over all n-sets, is then

$$\bar{j} = 2^{-n} J_n(1) = \sum_{k=1}^{n} (k-1) 2^{-k+1}$$

$$\sim \sum_{k=2}^{\infty} \frac{(k-1)}{2^{k-1}}$$

$$= 2 \qquad (n \to \infty).$$

In the case of n-permutations with k cycles, a similar calculation yields

$$J_n(x) = (n - 1 + x) J_{n-1}(x) + n \sum_{k=1}^{n-1} x^k,$$

whose solution is

$$J_n(x) = \{(x(x+1)\cdots(x+n-1)\}\sum_{k=1}^{n} k\left\{\frac{x+x^2+\cdots+x^{k-1}}{x(x+1)\cdots(x+k-1)}\right\}$$

and we find

$$\bar{j} = \sum_{k=1}^{n} \frac{k}{(k-1)!}$$

$$\sim 2e \qquad (n \to \infty).$$

For partitions of an n-set with k classes, the final result is

$$\bar{j} = \sum_{k=2}^{n} \frac{k^2}{2b_k}$$

$$\sim 2.79 \qquad (n \to \infty),$$

where the b_k are the Bell numbers.

6. The selection algorithm

We describe here a concrete implementation of these ideas for families which live on the lattice points of the plane.

A particular object is described by three arrays:

(μ_i, ν_i) $(i = 1, m)$, the vertex sequence along the path

edge (i) $(i = 1, m)$, the edge sequence, or codeword.

The algorithm assumes that $b(n, k)$, the number of objects of order (n, k), has been pre-computed for all relevant (n, k).

In general, from (μ, ν), we can go "West", to (x_w, ν), or "Southwest", to $(x_s, \nu - 1)$. The function x_w is $\mu - 1$ except for the partitions of n with largest part k, where it is $\mu - \nu$. The function x_s is $\mu - 1$ except for compositions of n into k parts, where it is μ.

If $\varphi(\mu, \nu)$ (resp. $\psi(\mu, \nu)$) are the number of Westbound (resp. Southwestbound) edges from (μ, ν), then we abbreviate $\varphi(\mu(j), \nu(j))$ (resp. $\psi(\mu(j), \nu(j))$ by φ_j (resp. ψ_j). Further, $b(x_w(\mu(j), \nu(j)), \nu(j))$ is $b_w(j)$ (the number of objects at the Western neighbor), and similarly for $b_s(j)$.

The specificity of the combinatorial family appears solely in the functions b_s, b_w, x_s, φ, ψ, and does not appear explicitly in the following algorithm, which is quite general. I have prepared from the algorithm a FORTRAN program of about 120 instructions which handles the seven families shown below.

1. k-subsets of an n-set,
2. partitions of an n-set into k classes,
3. permutations of n letters with k cycles,
4. vector k-subspaces of V_n over $GF(q)$,
5. permutations of n letters with k runs,

6. partitions of n with largest part k,
7. compositions of n into k parts.

Decoding is not included, but these algorithms are all simple and are described in [3]. One follows the path backwards from the terminal vertex to (n, k), on each edge carrying out the command given by the "road sign" on that edge.

Algorithm Select

(A) [Entry for random object] $r \leftarrow b(n,k)*$(random number)
(B) [Entry for unranking] $j \leftarrow 1$; $r' \leftarrow r$
(C) [Extend from j^{th} step] $(\mu_j, \nu_j) \leftarrow (n, k)$; $m \leftarrow j$
(D) [Reached terminal vertex?] If $\varphi_m + \psi_m = 0$, set $m \leftarrow m-1$ and return; if $r' \geq \varphi_m b_w(m)$, to (E); [Go West] edge $(m) \leftarrow \lfloor r'/b_w(m) \rfloor$; $r' \leftarrow r' -$ edge $(m) b_w(m)$; $(\mu_{m+1}, \nu_{m+1}) \leftarrow (x_w(m), \nu_m)$; $m \leftarrow m+1$; to (D).
(E) [Go Southwest] $r' \leftarrow r' - b_w(m)\varphi_m$; edge$(m) \leftarrow \varphi_m + \lfloor r'/b_s(m) \rfloor$; $r' \leftarrow r' - (\text{edge}(m) - \varphi_m) b_s(m)$; $(\mu_{m+1}, \nu_{m+1}) \leftarrow (x_s(m), \nu_m - 1)$; $m \leftarrow m+1$; to (D).
(F) [Entry for ranking] $r \leftarrow 0$;
For $j = 1, m-1$ **do**:
(F1) If $\nu_{j+1} \neq \nu_j$, to (F2); $r \leftarrow r + \text{edge}(j) b_w(j)$; next j.
(F2) $r \leftarrow r + \varphi_j b_w(j) + (\text{edge}(j) - \varphi_j) b_s(j)$; next j.
End
Exit
(G) [First entry for lex sequencing] $r \leftarrow 0$; to (B).
(H) [Later entries for lex sequencing] $j \leftarrow m$
(K) [Is j^{th} edge moveable?] If edge $(j) < \varphi_j + \psi_j - 1$, to (L); [Backtrack] $j \leftarrow j-1$; If $j \neq 0$, to (K); Final exit.
(L) [Augment edge] edge$(j) \leftarrow$ edge$(j) + 1$; $l \leftarrow j+1$; If edge$(j) \neq \varphi_j$, to (M); $(\mu_l, \nu_l) \leftarrow (x_s(l-1), \nu_l - 1)$
(M) [Extend with 0's to terminal vertex] $r' \leftarrow 0$; $m \leftarrow l$; to (D) □

Appendix 1

Table A1 below shows the 3-subsets of $\{1,2,3,4,5\}$, following lexicographic order of their codewords. Fig. A1 shows a portion of the relevant graph.

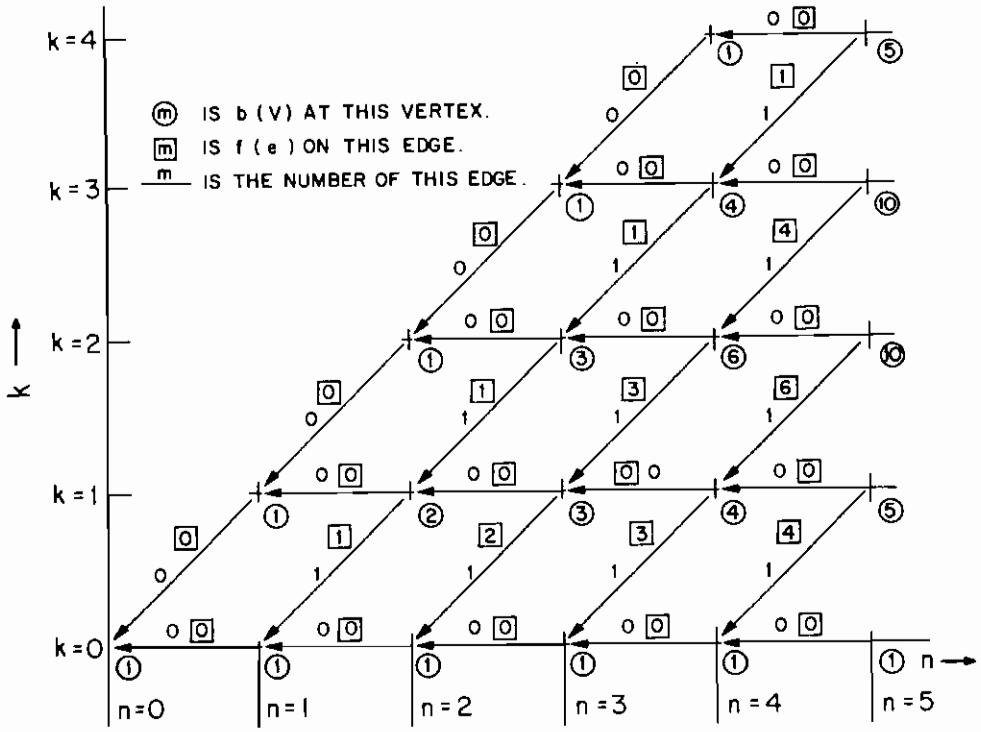

Fig. A1. The graph for k-subsets of n-sets.

Table A1

Rank	Codeword	Subset
0	00000	{1, 2, 3}
1	01000	{1, 2, 4}
2	01101	{1, 3, 4}
3	01110	{2, 3, 4}
4	10000	{1, 2, 5}
5	10100	{1, 3, 5}
6	10110	{2, 3, 5}
7	11000	{1, 4, 5}
8	11010	{2, 4, 5}
9	11100	{3, 4, 5}

Appendix 2

Table A2 below shows the partitions of $\{1, 2, 3, 4, 5\}$ into 3 classes, following lexicographic order of their codewords. Fig. A2 shows a portion of the relevant graph.

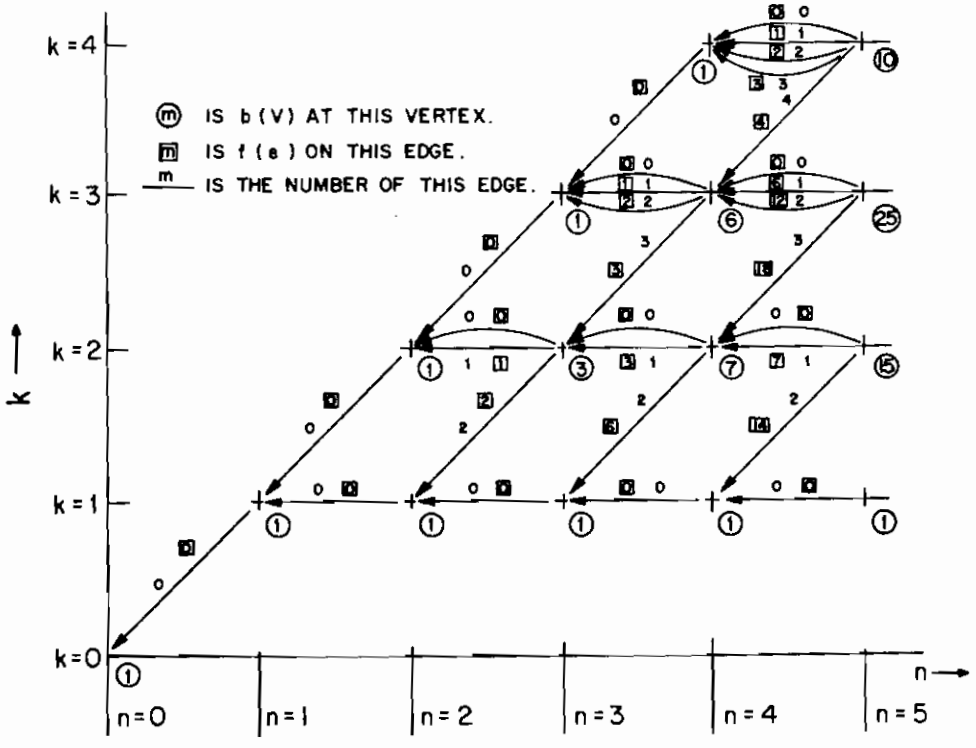

Fig. A2. The graph for partitions of an n-set into k classes.

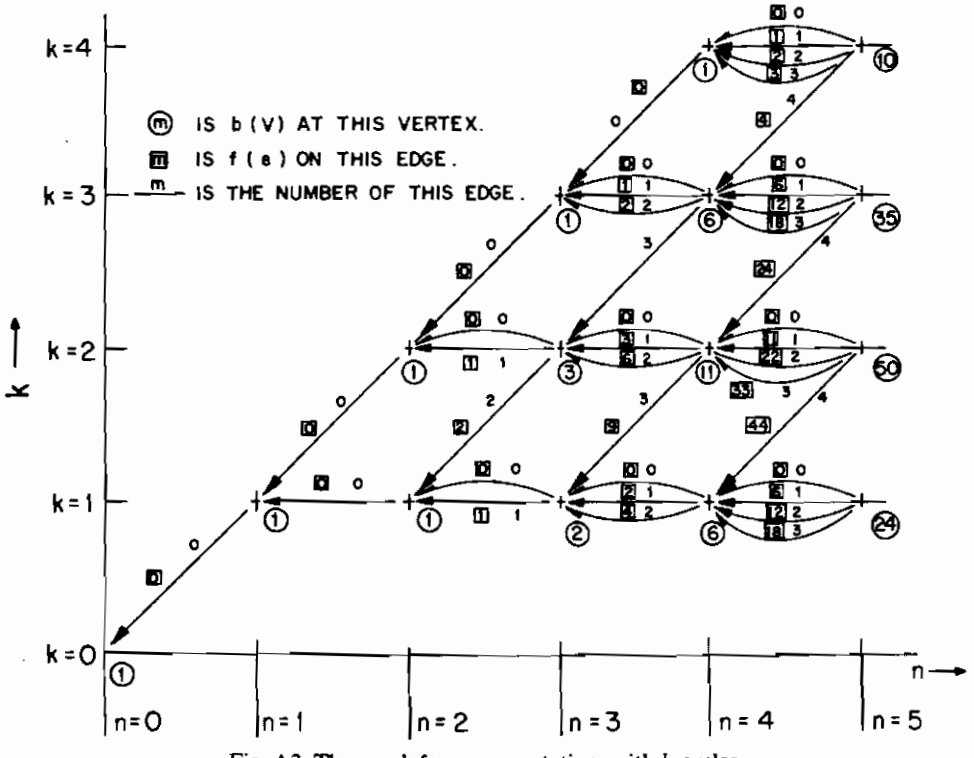

Fig. A3. The graph for n-permutations with k cycles.

Table A2

Rank	Codeword	Partition
0	00000	(145)(2)(3)
1	01000	(15)(24)(3)
2	02000	(15)(2)(34)
3	03000	(135)(2)(4)
4	03100	(15)(23)(4)
5	03200	(125)(3)(4)
6	10000	(14)(25)(3)
7	11000	(1)(245)(3)
8	12000	(1)(25)(34)
9	13000	(13)(25)(4)
10	13100	(1)(235)(4)
11	13200	(12)(35)(4)
12	20000	(14)(2)(35)
13	21000	(1)(24)(35)
14	22000	(1)(2)(345)
15	23000	(13)(2)(45)
16	23100	(1)(23)(45)
17	23200	(12)(3)(45)
18	30000	(134)(2)(5)
19	30100	(14)(23)(5)
20	30200	(124)(3)(5)
21	31000	(13)(24)(5)
22	31100	(1)(234)(5)
23	31200	(12)(34)(5)
24	32000	(123)(4)(5)

Appendix 3

Table A3 below shows the permutations of $\{1, 2, 3, 4, 5\}$ with 3 cycles, following lexicographic order of their codewords. Fig. A3 shows a portion of the relevant graph.

Table A3

Rank	Codeword	Partition
0	00000	(154)(2)(3)
1	01000	(15)(24)(3)
2	02000	(15)(2)(34)
3	03000	(153)(2)(4)
4	03100	(15)(23)(4)
5	03200	(152)(3)(4)
6	10000	(145)(2)(3)
7	11000	(1)(254)(3)
8	12000	(1)(25)(34)
9	13000	(135)(2)(4)
10	13100	(1)(253)(4)
11	13200	(125)(3)(4)
12	20000	(14)(25)(3)
13	21000	(1)(245)(3)
14	22000	(1)(2)(354)
15	23000	(13)(25)(4)
16	23100	(1)(235)(4)
17	23200	(12)(35)(4)
18	30000	(14)(2)(35)
19	31000	(1)(24)(35)
20	32000	(1)(2)(345)
21	33000	(13)(2)(45)
22	33100	(1)(23)(45)
23	33200	(12)(3)(45)
24	40000	(143)(2)(5)
25	40100	(14)(23)(5)
26	40200	(142)(3)(5)
27	41000	(134)(2)(5)
28	41100	(1)(243)(5)
29	41200	(124)(3)(5)
30	42000	(13)(24)(5)
31	42100	(1)(234)(5)
32	42200	(12)(34)(5)
33	43000	(132)(4)(5)
34	43100	(123)(4)(5)

References

[1] Paul Erdös and Joseph Lehner, The distribution of the number of summands in the partitions of a positive integer, Duke Math. J. 8 (1941) 335–345.
[2] A. Nijenhuis and H.S. Wilf, Combinatorial Algorithms (Academic Press, NY, 1975; 2nd ed., 1978).
[3] H.S. Wilf, A unified setting for sequencing, ranking and selection algorithms for combinatorial objects, Adv. Math. 24 (1977) 281–291.