and those of other researchers which complement the factorizations of $n^4 + 1$ published by Cunningham.

11 Rue Jean Jaurès
Luxembourg

1. A. J. C. CUNNINGHAM, *Binomial Factorisations*, v. I and IV, Francis Hodgson, London, 1923 (especially v. I, p. 113–119).
2. S. HOPPENOT, *Tables des Solutions de la Congruence* $x^4 \equiv -1 (mod\ N)$ *pour* $100000 < N < 200000$, Librairie du *Sphinx*, Brussels, 1935.
3. A. DELFELD, "Table des solutions de la congruence $X^4 + 1 \equiv 0 (\mathrm{mod}\ p)$pour $300000 < p < 350000$," Institut Grand-Ducal de Luxembourg, Section des Sciences, *Archives*, v. 16, 1946, p. 65–70.
4. A. GLODEN, "Table des solutions de la congruence $X^4 + 1 \equiv 0 (\mathrm{mod}\ p)$pour $2 \cdot 10^5 < p < 3 \cdot 10^5$," *Mathematica* (Rumania), v. 21, 1945; *Table des solutions de la congruence* $x^4 + 1 \equiv 0 (mod\ p)$ *pour* $350000 < p < 500000$, Centre de Documentation Universitaire, Paris, 1946; *Table des solutions de la congruence* $x^4 + 1 \equiv 0 (mod\ p)$ *pour* $500000 < p < 600000$, Luxembourg, author, rue Jean Jaurès 11, 1947; *Table des solutions de la congruence* $x^4 + 1 \equiv 0 (mod\ p)$ *pour* $600000 < p < 800000$, Luxembourg, published by the author, 1952; *Table des solutions de la congruence* $x^4 + 1 = 0 (mod\ p)$ *pour* $800000 < p < 1000000$, Luxembourg, published by the author, 1959.
5. M. KRAITCHIK, *Recherches sur la Théorie des Nombres*, v. 2, Gauthier-Villars, Paris, 1929, p. 116–117.
6. N. G. W. H. BEEGER, *Additions and corrections to "Binomial Factorisations" by Lt. Col. A. J. C. Cunningham*, Amsterdam, 1933.
7. A. GLODEN, "Compléments aux tables de factorisation de Cunningham," *Mathesis*, v. 55, 1945–46, p. 254–256; *ibid.*, v. 61, 1952, p. 49–50, 101, 305–306; v. 68, 1959, p. 172. See also *Intermédiaire des Recherches Mathématiques*, v. 4, 1948, p. 39.

On the Generation of All Possible Stepwise Combinations

By Gary Lotto

Conventionally, when all possible combinations of all possible subset sizes from a set of n are desired, a binary count is performed. Associating the units digit with the number 1, the two's digit with the number 2, the four's digit with the number 3, etc., the binary count 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, etc., becomes associated with the combinations 1, 2, 12, 3, 13, 23, 123, 4, etc. This is useful in such procedures as the analysis of variance.

The above order of combinations requires that, when computing on data from one combination to the next, either (a) the calculation starts anew, or (b) if algorithms exist for generating a new function from the old one by single steps of either including or deleting a number from the combination, more than one step may be required. For example, we may go from the combination "2" to the combination "12" by "including 1." But going from "12" to "3" requires "deleting 1, deleting 2, and including 3."

Given, then, that a problem may be solved for some combination of k elements from the solution for the superset of $(k + 1)$ elements or the subset of $(k - 1)$ elements, is there an algorithm for generating all possible combinations which goes through the fewest recursions?

Received May 18, 1961.

TABLE 1

i	$A(i)$	$B(i)$	$C(i)$
1	00001	+1	1
2	00010	+2	1 2
3	00011	−1	2
4	00100	+3	2 3
5	00101	+1	1 2 3
6	00110	−2	1 3
7	00111	−1	3
8	01000	+4	3 4
9	01001	+1	1 3 4
10	01010	+2	1 2 3 4
11	01011	−1	2 3 4
12	01100	−3	2 4
13	01101	+1	1 2 4
14	01110	−2	1 4
15	01111	−1	4
16	10000	+5	4 5

The author has used the following algorithm to generate all combinations of independent variables in a multiple regression problem:

(1) For each step, carry the cycle number i of the combination which is to be generated.

(2) Divide i by 2, then the quotient by 2, etc., until the remainder is not 0. The number of divisions performed is k, the number to be included or deleted.

(3) Divide the quotient of the last division in (2) by 2. If the remainder is 0, include. If the remainder is 1, delete.

The algorithm is equivalent to inspecting the lowest non-zero bit in the binary representation of i. If this is the kth bit (counting from the right), the number k is to be included or deleted. The $(k + 1)$st bit instructs inclusion or deletion: if 0, include; if 1, delete.

Define $A(i)$ as the binary representation of i, $B(i)$ as $+k$ if the number k is to be included, or $-k$ if k is to be deleted on cycle i, and $C(i)$ as the resultant combination. Table 1 gives the first 16 values of i and these functions.

Given combinations 1 through $(2^{k-1} - 1)$, all combinations of $(k - 1)$ elements, the additional combinations which must be generated in order to produce all combinations of k elements are reproductions of the first $(2^{k-1} - 1)$ combinations, to each of which has been added the kth element, plus the combination of element k alone (in effect, a reproduction of the zero combination, plus element k).

The algorithm produces these combinations by: (1) including k on the 2^{k-1}st cycle, and not deleting it before the 2^kth cycle, and (2) reproducing the $B(i)$'s in reverse order with opposite sign $(B(2^{k-1} + c) = -B(2^{k-1} - c))$, thus on each cycle deleting from the combination that which we, $2c$ cycles before, included into

it, or including that which we, $2c$ cycles before, deleted from it, until the (2^{k-1})st combination, which corresponds to the empty set plus element k.

Proof of (1). Since the binary representation of 2^{k-1} is a 1 bit followed by $(k-1)$ zeros, the kth element is included on cycle 2^{k-1}. The kth element will remain until the binary number 11 followed by $(k-1)$ zeros appears. This will be on cycle number $(2^k + 2^{k-1}) > (2^k - 1)$. Thus, all combinations from 2^{k-1} through $(2^k - 1)$ will include the kth element.

Proof of (2). Since $(2^{k-1} + c) + (2^{k-1} - c) = 2^k$, the binary representations of $(2^{k-1} + c)$ and $(2^{k-1} - c)$ correspond in all their low-order zeros, and the low-order 1, in which they also correspond. The bit above the 1 must differ in the two numbers, due to the binary carry. Thus, $B(2^{k-1} + c) = -B(2^{k-1} - c)$.

To complete the proof by induction, we may note, by Table 1, that the algorithm has generated all combinations for $k \leqq 4$.

University of Pittsburgh, and
American Institute for Research
Pittsburgh, Pa.

Generation of Permutations by Addition

By John R. Howell

1. **Introduction.** Suppose one wishes to generate the $k!$ permutations of k distinct marks. Representing these k marks by 0, 1, 2, $\cdots$, $(k-1)$ written side by side to form the "digits" of a base k integer, then the repeated addition of 1 will generate integers whose "digits" represent permutations of k marks. Many numbers are also generated which are not permutations. D. H. Lehmer [2] states that this so-called addition method can be made more efficient by adding more than 1 to each successive integer.

2. **Method.** In this note, we show that the correct number greater than 1 to add to this integer is a multiple of $(k-1)$ radix k.

LEMMA 1. *The arithmetic difference radix k between an integer composed of mutually unlike digits and another integer composed of a permutation of the same digits is a multiple of* $(k-1)$.

Considering the process of "casting out nines," it is obvious that the two integers are congruent mod $(k-1)$. Hence, their difference is zero mod $(k-1)$.

The method seems to have two advantages. First, one can generate all $k!$ permutations in lexicographic order. Second, all permutations "between" two given permutations can be obtained. The process can be made to be cyclic if upon obtaining $(k-1), \cdots, 0$ one takes the next permutation to be $0, 1, \cdots, (k-1)$.

3. **Example.** Suppose we wish to generate the 4! permutations of 4 marks. Representing these 4 marks by 0, 1, 2 and 3, we add 3 radix 4 to 0123 to get 0132. Continuing this process we get the 4! permutations desired. The array below shows

Received September 11, 1961.