A Comparison of Combination Generation Methods

SELIM G. AKL
Queen's University, Canada

It is empirically shown that the algorithm of Mifsud is the fastest existing combination generator This contradicts recent efficiency claims concerning other algorithms.

Key Words and Phrases algorithm, combinations
CR Categories· 5 30

INTRODUCTION

In his pioneering 1960 paper, Lehmer [6] discussed computer solutions to various combinatorial problems such as set inclusion, the evaluation of a function of several variables, and the generation of combinatorial objects. In particular, he presented an algorithm for generating all combinations of n objects taken k at a time (see also [7]). To our knowledge, at least seven other algorithms have since been described that perform the same basic task [1-5], [8-10].

Recently, Payne and Ives [11] provided a new implementation of the algorithm of Liu and Tang [8] and compared it with a few other combination generators. The main conclusion of their work is that the new implementation is "a most attractive combination generator because it possesses so many advantageous attributes." One very important "attribute" with which the authors of [11] claim their new implementation has endowed the algorithm of [8] is speed (of program execution). Although this is an undeniable fact, we believe that it does not in any way justify their conclusion. Indeed, as our experiments described in the next section show, several other combination generation algorithms are faster than the new method.

The algorithm of Mifsud [9], which we found to be the best in terms of speed (for all values of n and k tried), is very briefly referred to in [11]. Ironically, "speed" is not even mentioned by Payne and Ives among the attributes of Mifsud's algorithm, and the "operation-count" comparative analysis they report ignores the algorithm altogether. (Also omitted from the comparison are the

This work was supported by the National Sciences and Engineering Research Council of Canada under Grant NSERC-A3336

Author's address. Department of Computing and Information Science, Queen's University, Kingston, Ont., Canada K7L 3N6.

Table I. Running Times of the Various Algorithms on the Burroughs B6700/2 Computer

	Execution time[a] when n, k =				
Algorithm	10, 2	20, 6	20, 10	25, 8	25, 12
Bitner et al. [1]	0:0·0:0	0:0:6:15	0:0 24·28	0:2·38·6	0:12:57:34
Chase [2]	0 0 0:0	0 0:7·57	0·0·31:29	0:3:34:12	0:14·9:16
Ehrlich [4]	0:0:0·0	0:0:5:35	0:0:27.18	0 2:21:0	0:11:9:44
Kurtzberg [5]	0:0.0.0	0 0 4:27	0:0:27:14	0:2:8:33	0.10:34:58
Lehmer [6]	0 0.0·0	0.0:2:45	0:0:28:27	0:1.24.20	0:13:3:13
Liu and Tang [8]	0.0:0:0	0:0:8:23	0:0:42.15	0·4.28:0	0·17·18:24
Mifsud [9]	0:0:0:0	0·0 2 22	0 0:18:12	0:1:8:26	0:7:46:7
Nijenhuis and Wilf [10]	0:0·0·0	0:0:6:31	0:0:33:5	0:2·57:2	0:14:38:40
Payne and Ives [11]	0 0.0 0	0:0·7.15	0:0:28:22	0·3:3:7	0:14·31 26

[a] Given in hours : minutes : seconds · ticks.

Table II. Running Times of the Various Algorithms on the IBM 370/138 Computer

	Execution time[a] when n, k =				
Algorithm	10, 2	20, 6	20, 10	25, 8	25, 12
Bitner et al [1]	0.0.0:0	0:0:7.18	0:0.43.2	0:2:15:2	0:18:24:52
Chase [2]	0:0:0:0	0:0:8.19	0.0:48.40	0.3:2:5	0:18:22:30
Ehrlich [4]	0:0:0 0	0:0:6:28	0:0:44:49	0:2:16·38	0:14:29:58
Kurtzberg [5]	0:0:0:0	0:0:6:13	0:0:45:20	0:2:10:48	0:15:4:32
Lehmer [6]	0:0:0:0	0:0:4:4	0:0:45:26	0:1:48:49	0:18:25:12
Liu and Tang [8]	0:0:0:0	0:0:9:6	0:0:50:21	0:5:14:3	0:20:13:9
Mifsud [9]	0:0:0:0	0:0:3:50	0:0:37:59	0:1:20:9	0:8:3:18
Nijenhuis and Wilf [10]	0:0:0:0	0:0:7:25	0:0.49.15	0:2:32:36	0:20:13:9
Payne and Ives [11]	0:0·0·0	0:0·8:20	0·0:45:6	0:2·56·3	0.18:45:56

[a] Given in hours : minutes seconds . ticks.

algorithms of Kurtzberg [5], Lehmer [6], and Nijenhuis and Wilf [10]). It is the sole purpose of the present paper to rehabilitate this very efficient but almost forgotten combination generation method.

EMPIRICAL ANALYSIS

The nine algorithms appearing in [1, p. 520; 2; 4, p. 691; 5; 6, pp. 184–185; 8; 9; 10, pp. 30–31; and 11, p. 169] were all coded in FORTRAN and run on three different computers:

(1) the Burroughs B6700/2 (using the MCP release 3.0 FORTRAN IV compiler);
(2) the IBM 370/138 (using the OS/VS1 release 21.8-level FORTRAN G compiler);
(3) the DEC PDP 11/03 (using the FORTRAN IV compiler).

Tables I, II, and III contain the execution times of the programs running on the different machines for various values of n and k. Each table entry is in the form *hours : minutes : seconds : ticks.* The superiority of Mifsud's algorithm in terms of speed is obvious.

Table III. Running Tımes of the Various Algorithms on the PDP 11/03 Computer

	Execution time[a] when n, k =				
Algorithm	10, 2	20, 6	20, 10	25, 8	25, 12
Bitner et al. [1]	0:0:0:4	0:0 53:21	0:4:8:41	0:24:45:7	1.48:7·30
Chase [2]	0:0·0:5	0:1.1:2	0:4:30:10	0:28·8:11	1:57:56·40
Ehrlich [4]	0:0:0:4	0·0:53:50	0:4:28:43	0:23:45:15	1·52·35:47
Kurtzberg [5]	0·0:0·2	0:0:28:23	0:2:50:8	0:13·30:11	1:30:27·40
Lehmer [6]	0:0:0:1	0.0·18·48	0:3.58:48	0:9:33:53	1:58:59·16
Liu and Tang [8]	0·0·0:8	0:1:43 9	0:8:6.19	0.34·17:19	3:6:22:31
Mıfsud [9]	0:0:0 1	0:0:16:51	0:2:6 41	0:8:13:52	0·52:57:58
Nijenhuis and Wilf [10]	0:0·0:4	0:1.0:50	0·5:42:58	0 28 53:12	2:56:39·23
Payne and Ives [11]	0:0 0:4	0:1:3:20	0 5:10:14	0:28:58:7	2·5:6:24

[a] Given in hours : mınutes : seconds : ticks.

Table IV. Average Number of Operations Required by Each Algorıthm to Generate One Combination

Algorithm	$n\backslash k$	1	2	3	4	5	6	7	8
Bıtner et al. [1]	16	121.7	120 5	116.8	122.7	120.8	126 0	124.9	129.5
	32	201.8	200.3	196.3	203.3				
Chase [2]	16	70.0	53.4	45.7	41.7	39.1	37.5	36.6	36.1
	32	102.0	75.2	63.0	55.9				
Ehrlich [4]	16	35.9	31.3	30.8	31.1	31.4	31.8	32.2	32.7
	32	34.9	30.6	30.3	30.4				
Kurtzberg [5]	16	16.7	18.3	19.3	20.4	21.6	23.0	24.7	26.8
	32	16.9	17 7	18.0	18.5				
Lehmer [6]	16	5.6	7.1	8.6	10 3	12.6	15.8	20.6	28.5
	32	5.3	6.0	6.6	7 3				
Liu and Tang [8]	16	77.1	56.2	45.6	42.2	38.0	37 9	36.1	37.3
	32	132 8	92.3	72.0	62.3				
Mifsud [9]	16	4.6	6.7	8.4	10.2	12.3	14.8	17.6	21.1
	32	4.3	5.3	6.0	6.8				
Nijenhuis and Wilf [10]	16	16.9	17.9	19.3	21.3	21.9	24.7	25 2	28.7
	32	17.0	16 5	18.0	18.1				
Payne and Ives [11]	16	13.6	16.2	17.0	18.0	18.9	20.2	21.6	23.4
	32	13.3	15.6	15.9	16.3				

This finding was confirmed by a second experiment where an operation counter was inserted in each program and all operations (e.g., comparison, assignment, branching, arithmetic, logical) were equally weighed. The results are given in Table IV, which shows the average number of operations per combination for the same values of n and k used in [11]. The average was obtained by dividing the total number of operations required to generate the $\binom{n}{k}$ combinations by $\binom{n}{k}$.

CONCLUSION

Our results indicate that Mifsud's algorithm is the fastest available combination generator. This fact is confirmed by the consistent behavior of the program running on computers with different architectures. The use of this algorithm is therefore recommended for applications in which speed is an important factor. Interested readers are referred to [12] for a theoretical analysis of the algorithm.

REFERENCES

1. BITNER, J.R., EHRLICH, G., AND REINGOLD, E.M. Efficient generation of the binary reflected Gray code and its applications. *Commun. ACM 19*, 9 (Sept. 1976), 517-521
2. CHASE, P.J. Algorithm 382. Combinations of *M* out of *N* objects. *Commun. ACM 13*, 6 (June 1970), 368.
3. EHRLICH, G. Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. *J ACM 20*, 3 (July 1973), 500-513.
4. EHRLICH, G. Algorithm 466. Four combinatorial algorithms. *Commun. ACM 16*, 11 (Nov. 1973), 690-691.
5. KURTZBERG, J. Algorithm 94. Combination. *Commun ACM 5*, 6 (June 1962), 344.
6. LEHMER, D.H Teaching combinatorial tricks to a computer. *Proc. Symp. Applied Mathematics X*, R. Bellman and M. Hal, Jr (Eds), Amer. Math. Soc., Providence, R.I., 1960, pp. 179-193.
7 LEHMER, D.H. The machine tools of combinatorics. In *Applied Combinatorial Mathematics*, E. F Beckenbach (Ed.), Wiley, New York, 1964, pp 5-31.
8. LIU, C.N., AND TANG, D.T. Algorithm 452. Enumerating combinations of *m* out of *n* objects. *Commun. ACM 16*, 8 (Aug. 1973), 485.
9 MIFSUD, C.J. Algorithm 154. Combination in lexicographical order. *Commun. ACM 6*, 3 (Mar. 1963), 103
10. NIJENHUIS, A., AND WILF, H.S. *Combinatorial Algorithms.* Academic Press, New York, 1975, pp. 21-34.
11. PAYNE, W.H., AND IVES, F.M. Combination generators. *ACM Trans Math. Softw. 5*, 2 (June 1979), 163-172.
12. REINGOLD, E.M., NIEVERGELT, J., AND DEO, N. *Combinatorial Algorithms.* Prentice-Hall, Englewood Cliffs, N.J., 1977, pp. 179-181.

Received July 1979, revised July 1980; accepted August 1980.