STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING ZUIVERE WISKUNDE

ZW 1969 - 014

An ALGOL-60 Algorithm for the verification of a combinatorial conjecture on a finite Abelian group.

by

P. van Emde Boas

December 1969

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

§1 Introduction.

This report describes an Algorithm which verifies the following conjecture:

Let S be a sequence of length 18 of elements from the Abelian group $C_7 \oplus C_7$ such that S contains no non-empty subsequence of length ≤ 7 with sum zero. Then S consists of three distinct elements each taken 6 times.

This conjecture is the formulation for $p = 7$ of a general conjecture which can be formulated for all primes:

Let S be a sequence of length $3p - 3$ of elements from $C_p \oplus C_p$ such that S contains no non-empty subsequence of length $\leq p$ with sum zero. Then S consists of three distinct elements each taken $(p-1)$ times.

This statement describes a condition on a prime p which we denote by (C). See [1], §5. It is shown there that a $C_p \oplus C_p$-sequence of length $> 3p - 3$ always contains a subsequence with length $\leq p$ and sum zero.

For explicite definitions of the used notations see [1].

The main problem in this field of research can be described in the following way:

Let G be a finite Abelian group. There exists an unique representation of G in the form:

$$G = C_{d_1} \oplus C_{d_2} \oplus \ldots \oplus C_{d_k} \qquad \text{with } 1 < d_1 \mid d_2 \mid \ldots \mid d_k$$

The integer $\Lambda(G) = d_1 + d_2 + \ldots + d_k - k$ clearly is an invariant of the group G. Another invariant $\lambda(G)$ is defined the following way:

A finite sequence of elements from G (G-sequence) is called <u>primitive</u> if it contains no non-empty subsequence such that the sum of the

elements of this subsequence is zero (zero-subsequence).
We define $\lambda(G)$ to be the maximal length of a primitive G-sequence. If $\omega(G)$ is the order of G then we have always the inequalities:

$$\Lambda(G) \leq \lambda(G) \leq \omega(G) - 1.$$

There are many groups for which $\Lambda(G) = \lambda(G)$, for example all p-groups (where $\omega(G) = p^n$ for some prime p) and all groups of the type $C_a \oplus C_{ab}$. It was proved in 1969 that this equality is not generally true; for example $\lambda((C_2)^4 \oplus C_6) \geq \Lambda((C_2)^4 \oplus C_6) + 1$ (see [1]).

The condition (C) is used to prove the equality $\lambda = \Lambda$ for groups G of the types:

$$G = C_2 \oplus C_{2nm_1} \oplus C_{2nm_2} \text{ and } G = C_3 \oplus C_{6nm_1} \oplus C_{6nm_2}$$

where n is the product of powers of primes satisfying (C) and either $m_1 = 1$, $m_2 \in \mathbb{N}$ or $m_1 = p^r$, $m_2 = p^s$ for some prime p. These are the types V and IX as described in [1] §1 and [2] §1.

Condition (C) is trivially true for p = 2 and 3. For p = 5 verification by hand is possible but the verification was performed also by the computer using some preliminary form of the described ALGOL-60 program. This program which was designed to perform the verification for both p = 5 and p = 7 proved to be too "slow" for p = 7. Therefore a specialised program was written for p = 7. This latter program is discussed in this report.

§2 contains a detailed description of the used "forbidden region"-algorithm. §3 contains the text of the program and its output. The lay-out of the program has been provided by the ALGOL-editor program. See [3].

§2 The forbidden-region Algorithm.

In this Algorithm the group $C_7 \oplus C_7$ is treated like the 7×7-chessboard which is given in diagram 1 below. Diagram 2 gives an enumeration of some elements of $C_7 \oplus C_7$ which is used in the program.

$\begin{bmatrix}0\\6\end{bmatrix}$	$\begin{bmatrix}1\\6\end{bmatrix}$	$\begin{bmatrix}2\\6\end{bmatrix}$	$\begin{bmatrix}3\\6\end{bmatrix}$	$\begin{bmatrix}4\\6\end{bmatrix}$	$\begin{bmatrix}5\\6\end{bmatrix}$	$\begin{bmatrix}6\\6\end{bmatrix}$
$\begin{bmatrix}0\\5\end{bmatrix}$	$\begin{bmatrix}1\\5\end{bmatrix}$	$\begin{bmatrix}2\\5\end{bmatrix}$	$\begin{bmatrix}3\\5\end{bmatrix}$	$\begin{bmatrix}4\\5\end{bmatrix}$	$\begin{bmatrix}5\\5\end{bmatrix}$	$\begin{bmatrix}6\\5\end{bmatrix}$
$\begin{bmatrix}0\\4\end{bmatrix}$	$\begin{bmatrix}1\\4\end{bmatrix}$	$\begin{bmatrix}2\\4\end{bmatrix}$	$\begin{bmatrix}3\\4\end{bmatrix}$	$\begin{bmatrix}4\\4\end{bmatrix}$	$\begin{bmatrix}5\\4\end{bmatrix}$	$\begin{bmatrix}6\\4\end{bmatrix}$
$\begin{bmatrix}0\\3\end{bmatrix}$	$\begin{bmatrix}1\\3\end{bmatrix}$	$\begin{bmatrix}2\\3\end{bmatrix}$	$\begin{bmatrix}3\\3\end{bmatrix}$	$\begin{bmatrix}4\\3\end{bmatrix}$	$\begin{bmatrix}5\\3\end{bmatrix}$	$\begin{bmatrix}6\\3\end{bmatrix}$
$\begin{bmatrix}0\\2\end{bmatrix}$	$\begin{bmatrix}1\\2\end{bmatrix}$	$\begin{bmatrix}2\\2\end{bmatrix}$	$\begin{bmatrix}3\\2\end{bmatrix}$	$\begin{bmatrix}4\\2\end{bmatrix}$	$\begin{bmatrix}5\\2\end{bmatrix}$	$\begin{bmatrix}6\\2\end{bmatrix}$
$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}1\\1\end{bmatrix}$	$\begin{bmatrix}2\\1\end{bmatrix}$	$\begin{bmatrix}3\\1\end{bmatrix}$	$\begin{bmatrix}4\\1\end{bmatrix}$	$\begin{bmatrix}5\\1\end{bmatrix}$	$\begin{bmatrix}6\\1\end{bmatrix}$
$\begin{bmatrix}0\\0\end{bmatrix}$	$\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}2\\0\end{bmatrix}$	$\begin{bmatrix}3\\0\end{bmatrix}$	$\begin{bmatrix}4\\0\end{bmatrix}$	$\begin{bmatrix}5\\0\end{bmatrix}$	$\begin{bmatrix}6\\0\end{bmatrix}$

diagram 1

	29	30	31	32		
	25	26	27	28	33	34
	22	23	24	4	14	18
36	20	21	3	10	13	17
	19	2	7	9	12	16
35	1	5	6	8	11	15

diagram 2

The Algorithm choses a sequence S which contains no short zero-subsequence (a zero-subsequence of length ≤ 7). If the length is 18 the sequence is printed by OUTPUT. When the length is < 18 the Algorithm seeks whether or not there exists a group element by which S can be extended. If there exists no such element the last element of S is discarded and replaced by the next element in the enumeration of the group given in diagram 2 (if present). This way the program considers all possible sequences in a lexicographical order. The sequences themselves are ordered in such a way that the place of an element in the enumeration is not decreasing. This way out of any class of sequences which can be transformed into each other by permutation only the first in the lexicographical order is treated. After the last sequence has been discarded the Algorithm stops.

We now treat the details of the algorithm and the simplifications which are presupposed in the ALGOL-60 program.

2a: <u>The forbidden region.</u>

The elements by which the program permits S to be extended are numbered 1 upto 36 as is shown in diagram 2. The justification of this restriction will be given in section 2b treating the starting positions.

The elements can be reconstructed from their numbers by means of the arrays p and q which give the first resp. the second coordinate of the corresponding elements; the n-th element thus becomes $\begin{bmatrix} p[n] \\ q[n] \end{bmatrix}$.

The meaning of the array r is described in 2c.

A new element A must be chosen in such a way that no short zero-subsequences can be constructed from A together with elements already contained in S. This means that the element - A should not be the sum of a subsequence T of S of length ≤ 6.

These restrictions are implementated in the program by the creation of a "forbidden region" which describes at each moment the situation. This forbidden region is stored in the array k which is the one-dimensional implementation of a three-dimensional array K[0;6, 0:6, 4:21]. The correspondence between K and k is given by

$$K[a,b,c] = k[a + 7 * b + 49 * c]$$

The array K has to be considered as the union of $C_7 \oplus C_7$-chessboards for c = 4 upto 21. For every j = 4 ... 21 K[-, -, j] describes the restriction by the first j elements.

The integers K[x, y, j] are determined by the minimal length v of a subsequence of S with sum $\begin{bmatrix} 7-x \\ 7-y \end{bmatrix}$. For v > 6 we take K[x, y, j] = 0; else we take K[x, y, j] = 7 - v. If there exists no subsequence of S with sum $\begin{bmatrix} 7-x \\ 7-y \end{bmatrix}$ we take also K[x, y, j] = 0.

The restriction contained in the field $K[x, y, j]$ should be read as follows:

"The element $\begin{bmatrix} x \\ y \end{bmatrix}$ should not be reached in less than $K[x, y, j]$ steps"

or more explicitely:

If T is a sequence which contains a subsequence of length $\leq K[x, y, j]$ with sum $\begin{bmatrix} x \\ y \end{bmatrix}$ then $S \cup T$ contains a short zero-subsequence.

This means for example that $K[0, 0, j] = 7$ for every j.
If $K[x, y, j] \geq 1$ extension of S by $\begin{bmatrix} x \\ y \end{bmatrix}$ is impossible and forbidden in the program.

This property of the program is used in the elaboration of case B. Then we sometimes put $K[x, y, j] = 1$ artificialy in order to prevent the choice of the element $\begin{bmatrix} x \\ y \end{bmatrix}$. This trick has no influence on the recursive definition of $K[x, y, j]$ in terms of $K[-, -, j-1]$ which we describe next.

Let S be extended by an jth element say $\begin{bmatrix} a1 \\ a2 \end{bmatrix}$ then we define:

$$K[x, y, j] := \max\{K[x, y, j-1], K[v, w, j-1] - 1\}$$

where $v \equiv x + a1$ and $w \equiv y + a2 \pmod 7$.

The program must perform addition modulo 7. This addition is stored in the array o where we have $o[a + 7 * b] \equiv a + b \pmod 7$ for $0 \leq a,b \leq 6$. See START.

2b: The starting position.

By pure mathematical reasoning much information on the possible sequences S can be gathered. This knowledge is used to simplify the Algorithm.

A <u>line</u> in $C_7 \oplus C_7$ is a non trivial cyclic subgroup of $C_7 \oplus C_7$. There are exactly 8 lines in $C_7 \oplus C_7$.
A "base" for $C_7 \oplus C_7$ can be chosen by first chosing two distinct lines and next chosing in each of these base lines a non zero element. These two elements now may be represented by $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$.

The most important simplification presupposed in the program is the assumption that nearly always the elements in S taken from the base lines are equal. To see why this assumption is correct we consider first only C_7-sequences.

Two C_7-sequences are considered equivalent if they can be transformed in each other by premutation of the elements and/or multiplication (mod 7) of all elements by a fixed integer k $(1 \leq k \leq 6)$. For example the sequences (1,3,5) and (1,2,3) are equivalent by multiplication by 3.

Below in table 1 we give representatives T of all equivalence classes of primitive C_7-sequences. Next to them we write (if possible) another sequence T' of the same length and consisting of equal elements. For T' the following holds:

"If a is the sum of a subsequence $U' \subset T'$ then a is also the sum of a subsequence $U \subset T$. Furthermore the length of such a sequence U is at most the length of U'."

Let S be a sequence containing a subsequence U of type T of elements within our line. If we replace U by the sequence U' consisting of equal elements of the corresponding type T' we get a new sequence S'. Our claim is that if S' contains a short zero-subsequence then S contains a short zero-subsequence also. This can be derived as follows:

Let V' be a short zero-subsequence of S' and put $W' = V' \cap U'$. Now there exist a subsequence $W \subset U$ such that the sum of W is equal to the sum of W' and such that the length of W is at most the length of W'. Now $V = (V' \setminus W') \cup W$ is a short zero-subsequence of S.

Table 1. Types of primitive C_7-sequences.

	Type T							Type T'					
length 2	1	1						1	1				
	1	2						1	1				
	1	3											
	1	4					(eq. 1 2)	4	4				
	1	5					(eq. 1 3)						
length 3	1	1	1					1	1	1			
	1	1	2					1	1	1			
	1	1	3					1	1	1			
	1	1	4					4	4	4			
	1	2	2				(eq. 1 1 4)	1	1	1			
	1	2	3					1	1	1			
	1	3	5				(eq. 1 2 3)	5	5	5			
	1	4	4				(eq. 1 1 2)	4	4	4			
	1	4	5				(eq. 1 2 3)	4	4	4			
	1	5	5				(eq. 1 1 3)	5	5	5			
length 4	1	1	1	1				1	1	1	1		
	1	1	1	2				1	1	1	1		
	1	1	1	3				1	1	1	1		
	1	1	2	2				1	1	1	1		
	1	1	4	4			(eq. 1 1 2 2)	4	4	4	4		
length 5	1	1	1	1	1			1	1	1	1	1	
	1	1	1	1	2			1	1	1	1	1	
length 6	1	1	1	1	1	1		1	1	1	1	1	1

This makes it possible to perform the algorithm in two steps. First we search for the sequences S' containing no short zero-subsequences with the property that all subsequences in S' of elements within one line are of one of the types T'. Afterwards we search for each of the produced sequences S', from which sequences $S \quad S'$ could be derived. The latter step will be found superfluous as all the accepted sequences S' will prove to consist of three elements each taken six times. As the only primitive C_7-sequences of length 6 consist of a generator of C_7 taken 6 times we conclude that S' can not be derived from a sequence $S \neq S'$.

From table 1 we see that each collection of elements within one line which forms a primitive sequence of a type T has a corresponding sequence of type T' consisting of equal elements with only one exception. This exception is the case that T is of type $(1,3)$ or $(1,5)$, i.e. S contains exactly two elements within a line say x and y which satisfy a relation $x = y + y + y$ or $y = x + x + x$.

Now let S be a $C_7 \oplus C_7$-sequence of length 18. As $C_7 \oplus C_7$ contains 8 lines it follows that one of the following cases is present:

CASE A1: There are two lines each containing at least three elements.

We may assume (as indicated above) that these elements are equal. We chose them as base elements and we conclude that S contains a subsequence $\binom{1}{0}\ \binom{1}{0}\ \binom{0}{1}\ \binom{0}{1}$.

CASE A2: There exists one line containing at least 4 elements and there exists another line containing two elements in a situation of the type $1 \quad 1$, $1 \quad 2$ or $1 \quad 4$.

Again we may replace this sequence by a sequence in which these elements are equal and again we have a subsequence $\binom{1}{0}\ \binom{1}{0}\ \binom{0}{1}\ \binom{0}{1}$.

CASE B: There exists one line containing at least 4 elements and all other lines contain either one element or two elements in a situation of the type 1 3 or 1 5.

By a suitable choice of a base we have a subsequence
$\binom{1}{0}\ \binom{1}{0}\ \binom{1}{0}\ \binom{1}{0}\ \binom{0}{1}\ \binom{0}{3}$.
Further in the algorithm the choice of an element x as a member of S makes it possible to exclude as possible candidates for next choices the elements x, 2x and 4x.

CASE A1 and CASE A2 give the same starting position. CASE B gives another starting position but also a restriction by which the algorithm can be simplified.

In diagram 3 and 4 we design the starting positions as they are defined explicitely by the program at the places labeled START resp. CASE B. Integers outside the square diagrams indicate the number of the fields in the array k where the information is stored.

238	6	0	0	0	0	4	5	244
231	5	0	0	0	0	3	4	237
224	0	0	0	0	0	0	0	230
217	0	0	0	0	0	0	0	223
210	0	0	0	0	0	0	0	216
203	0	0	0	0	0	0	0	209
196	7	0	0	0	0	5	6	202

Diagram 3. CASE A

336	6	0	0	2	3	4	5	342
329	0	0	0	0	0	0	0	335
322	6	0	0	2	3	4	5	328
315	5	0	0	1	2	3	4	321
308	0	0	0	0	0	0	0	314
301	0	0	0	0	0	0	0	307
294	7	0	0	3	4	5	6	300

Diagram 4. CASE B

2c: The choice of the next element.

The algorithm considers the sequences in a lexicographical order. Mathematical reasoning makes it possible to discard an important collection of sequences as being equivalent to some sequence already considered. The leading principles are distinct in CASE A and CASE B.

CASE A.

In this case the elements 33 and 34 are impossible as they are already in the forbidden region at the start of the algorithm.

There is symmetry with respect to mirroring the main diagonal (i.e. interchanging of the two base elements). The elements are numbered in a way that 1 - 4 are on the main diagonal and 5 - 18 and 19 - 32 are mirror images: if $5 \leq j \leq 18$ and $\begin{bmatrix} a \\ b \end{bmatrix} \to j$ then $\begin{bmatrix} b \\ a \end{bmatrix} \to j + 14$.

This makes it very easy to transform a sequence S in its mirror image $\tilde{S}$. In order to prevent the work of treating both S and $\tilde{S}$ the algorithm contains a procedure, defined by the ALGOL text after the label KAPKAP which rejects a priori, if they are distinct, the sequence with the largest number of elements in 19 - 32.

The r-value of an element (array r) describes whether or not an element is on the main diagonal (r=0), on the right side (r=1) or on the left side (r=-1) of the main diagonal. The variable sim counts the sum of the r-values of the elements in S. The r-value of the candidate new element is stored in the variable h.

KAPKAP performs the following steps:

If sim + h > 0 the sequence is accepted for further tests.

If sim + h < 0 the sequence S is rejected.

We can reject this situation as we know that the mirror image will be accepted if it shows up. We reject at the same time all possible extensions which are justified as these extensions should have sim-values < 0 anyhow, for all the elements by which extension is possible are in 19 - 36.

If sim + h = 0 there are two possibilities:

i) All elements are from 1 - 4. Now the sequence is accepted (<u>if</u> y = j + 1 <u>then</u> <u>goto</u> TEST).

ii) There are as many elements from 5 - 18 as from 19 - 32 in S. Now S is compared with its mirror image $\tilde{S}$ lexicographically; S is accepted if S precedes $\tilde{S}$ and S is rejected if $\tilde{S}$ precedes S.

In this comparison the markers y and z are used; y marks the first element in S from 5 - 18 and z marks the first element in S from 19 - 32. If this comparison gives no result on the old elements of S alone the new candidate element is given the lowest value such that $\tilde{S}$ is not preceding S.

The markers y and z are changed if needed. If S contains no elements from 5 - 18 y is made length(S)+ 1. Analoguous z = length (S) + 1 if S contains no element from 19 - 32. (See the statements after NEXT, CONTINUE and COUNT.)

<u>CASE B</u>.

In this case we exclude all repetitions of elements as the corresponding sequences can be supposed to be treated already in CASE A. After having chosen the element x in S the elements x + x and x + x + x + x can be excluded also (see §2b).

This restriction explains the differences between the statements in CASE A and CASE B.

NEXT B differs from NEXT by the fact that the element is at least the successor of the latest element in S. This way repetitions become impossible.

In TEST B there is a statement which has the effect of excluding x + x and x + x + x + x as possible elements by putting them artificially in the forbidden region; the corresponding field in the array K is given value 1 if the value was zero, else the value remains unchanged.

2d: The expansion of S by base elements.

When all the possible extensions of S by non-base elements have been considered and discarded the algorithm tests whether or not the sequence S can be extended by base elements up to length 18.
This is performed by the statements after the labels COUNT and EXPANSION.

As explained in §2b we only need to consider expansion by base elements $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$.
The results of the program demonstrate that all accepted sequences contain each base element six times, so expansion by distinct base elements is useless.

In CASE B the expansion procedure can be simplified as extension by the element $\begin{bmatrix}0\\1\end{bmatrix}$ produces sequences already treated in CASE A.

The number of elements $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$ by which S can be extended is at most 8 resp. 2 in CASE A resp.CASE B. We therefore skip the expansion procedure if j = length (S) < 10 resp. 16.

The algorithm now performs the following steps:

1°) If continuing the algorithm should mean that the fourth element of the sequence should be changed the treating of CASE A is considered to be completed. The program then starts again at CASE B. Similar the program stops the execution of CASE B if the sixth element should be changed.

2°) If there are still elements in S which may be changed the program transforms the latest "fixed" element from S into a "variable" element of S. The values of sim, y and z are adapted.

3°) If the length of the remaining sequence together with the variable element is < 10 (16) the next element is chosen. (See CONTINUE.) If not the expansion procedure is started.

4^o) The value of T is increased by 1. (T counts the number of sequences tested by EXPANSION.)

5^o) b1 becomes the number of elements $\begin{bmatrix}1\\0\end{bmatrix}$ by which S can be extended.

6^o) b2 becomes the number of elements $\begin{bmatrix}0\\1\end{bmatrix}$ by which S can be estended. (Label A.)

7^o) If $j + b1 + b2 < 18$ the length 18 cannot be reached and S is rejected. (Label B.)

8^o) If $j + b1 \geq 18$ or $j + b2 \geq 18$ extension by elements $\begin{bmatrix}1\\0\end{bmatrix}$ or $\begin{bmatrix}0\\1\end{bmatrix}$ alone is sufficient. The resulting sequences are printed.

9^o) If the field K[1, 1, j] contains a value ≥ 2 the expansion by elements $\begin{bmatrix}0\\1\end{bmatrix}$ and $\begin{bmatrix}1\\0\end{bmatrix}$ simultaneously is impossible and the sequence is rejected. (label D.)
(This statement was introduced as the result of a test program showed that for nearly all sequences to be treated by EXPANSION this field contained a value ≥ 2.)

10^0) The procedures MENE and TEKEL are mirror images. The most efficient one of the two is chosen. They test by how many elements $\begin{bmatrix}1\\0\end{bmatrix}$ S can be extended after having been extended by f = 1, 2, ..., b2 elements $\begin{bmatrix}0\\1\end{bmatrix}$ and vice versa. This number is stored in the array-field t[f]. Every time the length 18 is realised the sequence is printed.

11^0) The sequence is discarded (<u>goto</u> CONTINUE).

In CASE B only the steps 1^o), 2^o), 3^o), 5^o) and 8^o) are performed.

2e: <u>The structure of the algorithm</u>.

Up to now the details of the program have been described. Next we indicate the structure of the program "in the large".

<u>The procedure OUTPUT</u>.

OUTPUT prints an accepted sequence. The following lines are given:

1°) A line containing two integers giving the number of times the base elements $\begin{bmatrix}1\\0\end{bmatrix}$ resp. $\begin{bmatrix}0\\1\end{bmatrix}$ are contained in S. (In CASE B this second number is zero.)

2°) A line containing some pars of integers. These pairs denote the remaining non-base elements of S. (In CASE B the elements $\begin{bmatrix}0\\1\end{bmatrix}$ and $\begin{bmatrix}0\\3\end{bmatrix}$ are printed also.)

3°) An empty line.

START: The starting position of CASE A is defined.

NEXT: Length S is increaed by one. The new candidate element becomes equal to the latest element in S.

KAPKAP: The sequence is rejected if symmetry-reasoning demands it. (See §2c)

TEST: If the candidate element is not in the forbidden region it is accepted as a fixed member of S. The forbidden region is redefined at the level j. A new element can be chosen (go to NEXT)

else we arive at

CONTINUE: The candidate element is replaced by the next one. Again symmetry has to be considered (go to KAPKAP).

If there is no next element we arive at

COUNT: If we are ready CASE B is started, else the sequence is shortened and we continue the program at EXPANSION resp. CONTINUE depending on the length of the sequence.

EXPANSION: See §2d.

CASE B: The starting position of CASE B is defined.

NEXT B, TEST B, CONTINUE B, COUNT B and EXPANSION B perform the same function as the corresponding parts of the program in CASE A.

Function of the variables.

The arrays S and n contain the elements of the sequence S. S contains the accepted elements but n contains also the candidate element.

a1 and a2 become the coordinates of a candidate new element and are used for calculations.

The variables m and mm are used to indicate places in the array k that should have to be calculated each time again. c, l and x are used to store results of incomplete calculations which should have to be stored in arrays else.

j is the length of the sequence S together with the new element and i is the length of the accepted elements from S. We always have i = j - 1. Their values are increased by 1 at NEXT and decreased by 1 at COUNT.

e, f, g are controled variables in for-statements.

The other variables have been explained elsewhere.

The output procedures EXIT, NLCR, ABSFIXT, SPACE and PRINTEXT are machine-code-written procedures of the MC-ALGOL 60 system for the EL X-8 computer which are available without declarations. See [4].

```
begin comment Verification of condition ( C) for p=7 , revised
      program ,free of symmetry-repeatments ,with one-dimensional
      arrays testing both cases.PEVEB 17-7-69;
      integer array k[196:1077], S, n[4:21], o[0:48], t[0:6], p, q,
      r[1:36];
      integer a1, a2, b1, b2, c, e, f, g, h, i, j, l, m, mm, sim, x,
      y, z, T;

      procedure OUTPUT(aa, bb, cc); integer aa, bb, cc;
      begin NLCR; ABSFIXT(2, 0, 2 + bb); SPACE(4);
        ABSFIXT(2, 0, 2 + cc); NLCR;
        for g:= 5 step 1 until aa do
        begin ABSFIXT(2, 0, p[S[g]]); ABSFIXT(2, 0, q[S[g]]);
            SPACE(4)
        end;
        NLCR;
      end;

START: for f:= 0 step 1 until 6 do
      begin mm:= 7 × f;
        for e:= 0 step 1 until 6 do
        begin o[e + mm]:= if e + f < 7 then e + f else e + f - 7;
            k[e + mm + 196]:= 0
        end
      end;
      k[196]:= 7; k[238]:= k[202]:= 6; k[231]:= k[244]:= k[201]:= 5;
      k[243]:= k[237]:= 4; k[236]:= 3; S[4]:= 1; sim:= 0; j:= 4;
      z:= y:= 5; T:= 0;
      for e:= 1, 5, 6, 8, 11, 15 do q[e]:= 1;
      for e:= 2, 7, 9, 12, 16, 19 do q[e]:= 2;
      for e:= 3, 10, 13, 17, 20, 21 do q[e]:= 3;
      for e:= 4, 14, 18, 22, 23, 24 do q[e]:= 4;
      for e:= 1 step 1 until 4 do
      begin q[e + 24]:= 5; q[e + 28]:= 6; p[e]:= e; r[e]:= 0 end;
      for e:= 5 step 1 until 18 do
      begin p[e]:= q[e + 14]; r[e]:= 1; p[e + 14]:= q[e];
        r[e + 14]:= - 1
      end;
      n[4]:= 1;
NEXT: i:= j; j:= j + 1; n[j]:= x:= n[i]; h:= r[x];
      y:= y + 1 - abs(h); z:= z + sign(h + 1);
KAPKAP: if sim + h < 0 then goto COUNT; if sim + h = 0 then
      begin if y = j + 1 then goto TEST; c:= z - y;
        for e:= y step 1 until z - 2 do
        begin if S[e] > S[e + c] - 14 then goto COUNT;
            if S[e] + 14 < S[e + c] then
            begin sim:= 0; goto TEST end
        end;
        if S[z - 1] + 14 > x then n[j]:= x:= S[z - 1] + 14; sim:= 0;
        goto TEST
      end;
      sim:= sim + h;
TEST: mm:= 49 × i; if k[p[x] + 7 × q[x] + mm] = 0 then
      begin S[j]:= x; a1:= p[x]; a2:= q[x];
```

```
        for e:= 0 step 1 until 6 do
        begin c:= o[a1] + 7 × e]; m:= e + mm − 7;
           for f:= 0 step 1 until 6 do
           begin g:= k[c + 7 × o[a2 + 7 × f] + mm] − 1; m:= m + 7;
              l:= k[m]; k[m + 49]:= if g > l then g else l
           end
        end;
        goto NEXT
      end;
CONTINUE: x:= n[j]:= n[j] + 1; if x = 33 then goto COUNT;
      sim:= sim − r[x − 1]; h:= r[x]; if x = 5 then y:= j;
      if x = 19 then z:= j; goto KAPKAP;
COUNT: if i = 4 then
      begin NLCR; NLCR; PRINTTEXT(
⊀ these are all sequences of lenght 18 without short zerosubsequences CASE A ⊁
        ); NLCR; PRINTTEXT(
        ⊀ number of sequences checked by EXPANSION⊁);
        ABSFIXT(6, 0, T); goto CASEB
      end;
      if y = j + 1 then y:= j; if z = j + 1 then z:= j;
      sim:= 2 × z − y − j; j:= j − 1; i:= i − 1;
      if j < 10 then goto CONTINUE; T:= T + 1;
EXPANSION: m:= j × 49;
      for e:= 1 step 1 until 6 do
      begin if k[m + e] > e then
        begin b1:= e − 1; goto A end
        else b1:= e
      end;
A:    for f:= 1 step 1 until 6 do
      begin if k[m + 7 × f] > f then
        begin b2:= f − 1; goto B end
        else b2:= f
      end;
B:    if j + b1 + b2 < 18 then goto CONTINUE;
      if j + b1 > 18 then OUTPUT(j, b1, 0);
      if j + b2 ≥ 18 then OUTPUT(j, 0, b2);
D:    if k[m + 8] > 2 then goto CONTINUE;
      if b1 < b2 then goto MENE else goto TEKEL;
MENE: t[0]:= c:= b2;
      for e:= 1 step 1 until b1 do
      begin for f:= 1 step 1 until 6 do
        begin if k[m + e + 7 × f] > e + f then
           begin t[e]:= f − 1; goto E end
           else t[e]:= f
        end;
      E:  c:= if t[e] < c then t[e] else c;
        if j + e + c ≥ 18 then OUTPUT(j, e, c);
      end;
      goto CONTINUE;
TEKEL: t[0]:= c:= b1;
      for f:= 1 step 1 until 6 do
      begin mm:= 7 × f;
        for e:= 1 step 1 until 6 do
        begin if k[m + e + mm] > e + f then
           begin t[f]:= e − 1; goto F end
```

```
        else t[f]:= e
      end;
    F:  c:= if t[f] < c then t[f] else c;
      if j + c + f ≥ 18 then OUTPUT(j, c, f);
    end;
    goto CONTINUE;
CASEB: q[33]:= q[34]:= p[33]:= 5; p[34]:= 6; p[35]:= p[36]:= 0;
    q[35]:= 1; q[36]:= 3; S[5]:= 35; S[6]:= 36;
    for e:= 294 step 1 until 342 do k[e]:= 0;
    k[300]:= k[322]:= k[336]:= 6;
    k[299]:= k[315]:= k[328]:= k[342]:= 5;
    k[298]:= k[321]:= k[327]:= k[341]:= 4;
    k[297]:= k[320]:= k[326]:= k[340]:= 3;
    k[319]:= k[325]:= k[339]:= 2; k[318]:= 1; k[294]:= 7; n[6]:= 0;
    j:= 6; T:= 0;
NEXTB: i:= j; j:= j + 1; n[j]:= x:= n[i] + 1;
    if x = 35 then goto COUNTB;
TESTB: mm:= 49 × i; if k[p[x] + 7 × q[x] + mm] = 0 then
    begin S[j]:= x; a1:= p[x]; a2:= q[x];
      for e:= 0 step 1 until 6 do
      begin c:= o[a1 + 7 × e]; m:= e + mm - 7;
        for f:= 0 step 1 until 6 do
        begin g:= k[c + 7 × o[a2 + 7 × f] + mm] - 1; m:= m + 7;
          l:= k[m]; k[m + 49]:= if g > l then g else l
        end
      end;
      for e:= 1, 2 do
      begin a1:= o[a1 + 7 × a1]; a2:= o[a2 + 7 × a2];
        m:= mm + 49 + a1 + 7 × a2;
        k[m]:= if k[m] = 0 then 1 else k[m];
      end;
      goto NEXTB
    end;
CONTINUEB: x:= n[j]:= n[j] + 1; if x ≠ 35 then goto TESTB;
COUNTB: if i = 6 then
    begin NLCR; NLCR; PRINTTEXT(
      ≮ case B completed. Number of sequences type B found ≯);
      ABSFIXT(6, 0, T); if T = 0 then
      begin NLCR; PRINTTEXT(
≮ there are no sequences of type B. condition (C) is verified .≯
        ); EXIT
      end
    end;
    j:= j - 1; i:= i - 1; if j < 16 then goto CONTINUEB;
EXPANSIONB: m:= j × 49;
    for e:= 1 step 1 until 6 do
    begin if k[m + e] > e then
      begin b1:= e - 1; goto EB end
      else b1:= e
    end;
EB:  if j + b1 < 18 then goto CONTINUEB; T:= T + 1;
    OUTPUT(j, b1 + 2, - 2); goto CONTINUEB;
end
```

<u>Output of the program</u>.

6		6									
1	1	1	1	1	1	1	1	1	1	1	1
6		6									
2	1	2	1	2	1	2	1	2	1	2	1
6		6									
3	1	3	1	3	1	3	1	3	1	3	1
6		6									
4	1	4	1	4	1	4	1	4	1	4	1
6		6									
4	3	4	3	4	3	4	3	4	3	4	3
6		6									
5	1	5	1	5	1	5	1	5	1	5	1
6		6									
6	1	6	1	6	1	6	1	6	1	6	1

these are all sequences of length 18 without short zero-subsequences
CASE A.
Number of sequences checked by EXPANSION 4705.

CASE B completed. Number of sequences type B found 0.
There are no sequences of type B. Condition (C) is verified.

The program was executed 18th July 1969 by the EL-X-8 computer of the Mathematical Centre in Amsterdam. The execution took 123 mh form which 120 mh was needed for CASE A alone.

In April 1968 a test program performing all the steps in the program up to Label D and printing the accepted sequences together with the corresponding "chessboard"-diagrams describing the forbidden region at the level of the length of S was elaborated. Its results seemed to give already a verification of condition (C) for $p = 7$.
On resuming the research in July 1969 it was found that the "impossible" case B was not covered by this preliminary program and the presented program was written in order to fill this gap.

References.

[1] P. van Emde Boas — A combinatorial problem on finite Abelian groups II. Math. Centre Report ZW-1969-007.

[2] P.C. Baayen, P. van Emde Boas and D. Kruyswijk — A combinatorial problem on finite Abelian groups III. Math. Centre Report ZW-1969-008.

[3] H.L. Oudshoorn, H.N. Glorie and G.C.J.M. Nogarede — ALGOL editor. A program for standardizing program lay-out. Math. Centre Report MR 98 (Sept.1968).

[4] F.E.J. Kruzeman Aretz — Het MC-ALGOL 60-systeem voor de X-8. Math. Centre Report MR 81 (Dutch).