3. J. A. Serret, Cours du calcul différentiel et intégral, t. I, Gauthier-Villars, 1900, pp. 86–87.
4. E. Netto, Lehrbuch der Combinatorik, Teubner, Leipzig, 1927, p. 59.
5. Barnard and Child, Higher Algebra, Macmillan, New York, 1936, p. 36, or H. S. Hall, Higher Algebra, Macmillan, 1923, p. 170, or v. Mangoldt-Knopp, Einführung in die höhere Mathematik, Band I, Hirzel Verlag, Basel, 1954, p. 35.

A NEW VERSION OF THE EUCLIDEAN ALGORITHM

W. A. Blankinship, Department of Defense

Introduction. Given a set of positive integers (or elements of a Euclidean ring), $a_1, a_2, \cdots, a_n$, the Euclidean algorithm provides a method for computing the greatest common divisor, d, of these numbers. If the steps performed during the operation of the algorithm are traced back, it is possible to deduce elements $x_1, x_2, \cdots, x_n$ such that

$$d = a_1x_1 + a_2x_2 + \cdots + a_nx_n.$$

In nearly all applications, e.g., for the Chinese problem of remainders (see [1]), for finding the inverse of an element of a Galois field, etc., it is desired to find the elements x_i. Although the process of untangling the steps of the algorithm to find the x_i is straightforward, anyone who has ever tried it will appreciate the difficulty of deriving the necessary formulas and carrying them out without error. This paper sets forth an algorithm which, although equivalent to the Euclidean, is much easier to visualize, is easily programmed either on paper or on a computer, and, in addition, produces the x's.

The Algorithm. We make use of the trick, well-known to the computing trade, of carrying along a matrix to keep track of the operations which have been performed. To prepare for the algorithm we first form an n by $n+1$ matrix whose first column consists of the positive integers $a_1, a_2, \cdots, a_n$ and the rest of which is the identity matrix, as follows:

$$\begin{array}{ccccccc}
a_1 & 1 & 0 & 0 & \cdots & 0 \\
a_2 & 0 & 1 & 0 & \cdots & 0 \\
a_3 & 0 & 0 & 1 & \cdots & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdots & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdots & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdots & \cdot \\
a_n & 0 & 0 & 0 & \cdots & 1.
\end{array}$$

The algorithm consists of performing elementary row operations on this matrix so as to reduce all but one of the elements in the first column to zero. If we refer to the first element of a row (at any stage of the process) as the leader of that row, the algorithm may be formalized as follows:

Step 1. Select the row with the smallest nonzero leader and call it the "operator."

Step 2. Select any other row with a nonzero leader and call it the "operand." (When no operand can be found the process is completed.)

Step 3. Divide the leader of the operator into the leader of the operand, ignoring the remainder. Denote the quotient by q.

Step 4. Subtract q times the operator from the operand, recording the result as a new row and striking out the operand.

Step 5. Return to step 1.

When the process is completed (see step 2), the one remaining row whose leader is not zero will be

$$d, x_1, x_2, x_3, \cdots, x_n,$$

where

$$d = a_1x_1 + a_2x_2 + \cdots + a_nx_n.$$

We now verify the algorithm. That the process terminates is easily seen by noting that every time step 4 is performed, a column leader decreases, but never becomes negative. Hence the sum of the column leaders is a strictly decreasing positive integer. Since it cannot decrease more than $\sum a_i$ times, the process must terminate.

We next note that elementary row operations (such as step 4) preserve the greatest common divisor of the leaders (or of any column); that is, g.c.d.$(b_1, b_2, \cdots, b_n)$=g.c.d.$(b_1+\alpha b_j, b_2, b_3, \cdots, b_n)$ for any integer α and any $j \leqq n$ different from 1. When the last step is reached all the leaders are zero except that of the previous operand and that number must be the g.c.d. of the original set of leaders.

If we denote the matrix by (a, I) where a is a column vector consisting of the a_i, and I is the identity matrix, then since each elementary operation (including permutation of rows) is equivalent to multiplying by a nonsingular matrix, M_i, we have

$$\begin{aligned}\text{Final matrix} &= \cdots M_3M_2M_1(a, I)\\ &= (Ma, M),\end{aligned}$$

where M denotes the product of the matrices M_i. Thus the last n columns of the final matrix consist of the matrix M, and if the nonzero leader d occurs in say the jth row, and we let m_j be the jth row of M, then the above equation implies that

$$d = m_ja,$$

which was our contention. This completes the proof that the algorithm works. It is also useful to remember that a row of the matrix, at any stage of the algorithm, represents a linear equation relating the leader of that row to a linear combination of the original a's.

The following remarks, though mostly obvious, may help clarify what is going on, and make it evident that in the first column we are simply executing the Euclidean algorithm.

a. In step 1 the operator will always be the new row adjoined during the previous pass, unless its leader was zero.

b. In step 2, it will usually be most convenient to select the previous operator as the new operand.

c. In step 4, the leader of the new row will always be the remainder resulting from the division in step 3.

d. The process will still work for any selection of operand in step 1, provided only that its leader is not zero and that there is a row available with a larger leader. (Sometimes the work can be shortened by exercising judgment here).

e. In step 4, although the quotient q is the optimum multiplier to use and guarantees convergence, a different choice of multiplier does not cause an error, but perhaps lengthens the process.

An Example. We illustrate with $n=3$, $a_1=99$, $a_2=77$, $a_3=63$, and of course, $d=1$. In the following chart we have numbered the rows chronologically and show, for expository purposes, the ordinal numbers of the rows used to compute each new row. Instead of striking out old operands we have written x's in the column marked "validity."

Row	*Operator*	*Operand*	q	*Validity*				
1				x	99	1	0	0
2				x	77	0	1	0
3				x	63	0	0	1
4	3	2	1	x	14	0	1	−1
5	4	3	4	x	7	0	−4	5
6	5	4	2		0	0	9	−11
7	5	1	14		1	1	56	−70
8	7	5	7		0	−7	−396	495

Row 7 is to be interpreted to mean

$$1 = 1 \times 99 + 56 \times 77 - 70 \times 63,$$

as indeed it is.

Application to the Problem of Chinese Remainders. Given the residues $m_1, m_2, \cdots, m_n$ of a number, m, modulo a number of distinct primes, $p_1, p_2, \cdots, p_n$, respectively, it is required to find m. If we take $a_i = p_i^{-1}\Pi_1^n p_j$ and solve for the x_i as before, then $d=1$ and we have

$$1 = a_1x_1 + a_2x_2 + \cdots + a_nx_n.$$

Multiplying this equation by m and noting that $a_im = a_im_i$ (modulo Πp_j), we ob-

tain

$$m = a_1x_1m_1 + a_2x_2m_2 + \cdots + a_nx_nm_n \text{ (modulo } \Pi p_j).$$

In this case the value of x_i need only be known modulo p_i and this fact may be used throughout the algorithm by reducing all numbers in a given column modulo the appropriate prime. When this is done, an additional check on the calculations is provided in that any row with a zero leader must consist entirely of zeroes.

The example in the last section illustrates the Chinese problem of remainders when $p_1=7$, $p_2=9$, $p_3=11$. The result

$$1 = 99 + 56 \times 77 - 70 \times 63$$

implies that

$$1 = 99 + 2 \times 77 + 7 \times 63 \text{ (modulo 693)}$$

or

$$m = 99m_1 + 154m_2 + 441m_3 \text{ (modulo 693)}.$$

Now if we wished to know the smallest number m whose remainders modulo 7, 9, and 11 are, say, 6, 2, and 5, respectively, we simply substitute in the above formula:

$$m = 99 \times 6 + 154 \times 2 + 441 \times 5 = 3107 = 335 \text{ (modulo 693)}.$$

Application to Polynomial Rings. The algorithm works equally well for any Euclidean ring provided we interpret "smaller" and "larger" in terms of the ring's norm. Thus for polynomials over a field we would rephrase step 1 to read "Select the row with the leader of lowest degree and call it the operator." In addition, due to the freedom in the choice of a multiplier for step 4 (as noted in remark e), we can incorporate the division algorithm into our process, simply by replacing step 3 by "Divide the *leading term of the leader* of the operator into the *leading term of the leader* of the operand, calling the result q." Thus q will always be a monomial. If we represent a polynomial by the vector consisting of its coefficients, the algorithm goes very nicely, particularly if the coefficients are in GF (2), since step 4 will consist of merely shifting and adding.

The reader is invited to apply the algorithm to the following problem: Find the inverse of the polynomial x^3+x+1 modulo x^5+x^2+1, where the coefficients are taken modulo 2. Note that if these polynomials are relatively prime, the algorithm yields polynomials $p(x)$ and $q(x)$ such that $p(x)(x^3 + x + 1) + q(x)(x^5+x^2+1) = 1$. Reading this modulo x^5+x^2+1 yields the interpretation

$$p(x) = (x^3 + x + 1)^{-1} \text{ modulo } x^5 + x^2 + 1.$$

Reference

1. L. E. Dickson, History of the Theory of Numbers, 3 vols., Washington, 1919–1927; vol. II, pp. 57–67.