On invariants of G.C.D. algorithms

Nieuw Archief voor Wiskunde (3) I 105—112 (1953)

ON INVARIANTS OF G. C. D. ALGORITHMS

BY

N. G. DE BRUIJN and W. M. ZARING

§ 1. *Introduction.* Let a_1, a_2 be positive integers, $a_1 > a_2 > 0$. A set of equations of the form [1])

$$\begin{cases} a_1 = a_2 q_1 + \varepsilon_1 a_3 \\ a_2 = a_3 q_2 + \varepsilon_2 a_4 \\ \cdots\cdots\cdots\cdots\cdots \\ a_{n-1} = a_n q_{n-1} + \varepsilon_{n-1} a_{n+1} \\ a_n = a_{n+1} q_n \end{cases} \tag{1.1}$$

where

$$\begin{cases} a_1 > a_2 > a_3 > \ldots > a_n > a_{n+1} > 0 \\ \varepsilon_1 = \pm 1, \ \ldots, \ \varepsilon_{n-1} = \pm 1 \end{cases} \tag{1.2}$$

is an algorithm for finding the G.C.D. of a_1, a_2 (shortly: an *algorithm* [2]) for a_1, a_2). The number n is called the *length* of the algorithm. If a_1 is not a multiple of a_2, the first equation leaves two possibilities for a_3, etc. Hence several [3]) different algorithms exist for the same pair a_1, a_2, and it is natural to ask for *invariants*: any expression in terms of n, $a_1, \ldots, a_{n+1}$, $q_1, \ldots, q_n$, $\varepsilon_1, \ldots, \varepsilon_{n-1}$ is called an invariant whenever its value is the same for all algorithms for a_1, a_2. For instance, a_{n+1} is an invariant (its value being the G.C.D.), but n is not. Several invariants will be derived in the next sections.

An algorithm for a_1, a_2 is uniquely determined if a special condition is added, which determines at each step which value of ε_i has to be taken. Such a condition, of a general type, is the following

[1]) a, b, q, p, r, ε denote integers; x, y stand for rational numbers; θ (in § 4) is the only irrational occurring in the paper.

[2]) With the usual meaning the word algorithm indicates a device for constructing a set of equations. We do not use it in this sense, as we use it to indicate the set of equations itself.

[3]) In fact, it is not difficult to show, by induction, that the number of different algorithms for the pair a_1, a_2 equals a_2/d, where d is the G.C.D. of a_1 and a_2.

one. Let S be a subset of the set of rationals in $-1 < x < 1$, such that for each rational y the congruence $x \equiv y \pmod 1$ has exactly one solution $x \in S$. Now the condition to be imposed on ε_i is, that the remainder $\varepsilon_i a_{i+2}$ has to be such that $(\varepsilon_i a_{i+2}/a_{i+1}) \in S$ $(i = 1, \ldots, n-1)$. The algorithm satisfying this condition can be called the *S-algorithm* for a_1, a_2.

Well-known examples are the Euclidean Algorithm (E.A.), and the Least Remainder Algorithm (L.R.A.). For the E.A., S has to be the set $0 \leq x < 1$, which simply means that all ε's are $+1$. For the L.R.A., S has to be given by $-\frac{1}{2} < x \leq \frac{1}{2}$. We shall also consider the case where S is given by $-1 < x \leq 0$ (this can be called the Anti Euclidean Algorithm (A.E.A.)), and the case where S consists of the intervals $-1 < x \leq -\frac{1}{2}$, $\frac{1}{2} < x < 1$ and of the point $x = 0$ (the Greatest Remainder Algorithm (G.R.A.)).

KRONECKER proved that no algorithm for a_1, a_2 is shorter than the L.R.A. (see [2]). GOODMAN and ZARING [1] showed that the length of the E.A. can be calculated once the L.R.A. is known: the length of the E.A. exceeds the length of the L.R.A. by the number of negative ε's occurring in the L.R.A. Conversely they showed that the length of the L.R.A. can be found once the E.A. is known (for this case their result was more complicated). In the present paper we show that the length of the E.A., L.R.A., G.R.A. and A.E.A. for a_1, a_2 can be obtained from any arbitrary algorithm for a_1, a_2. A few complementary results will follow immediately:

1. No algorithm is shorter than the L.R.A. (Kronecker's theorem), and we can describe exactly by which choices of the ε's algorithms are obtained which have the same length as the L.R.A.

2. The G.R.A. is longer than any other essentially different algorithm.

The results of [1] were established by a method using construction of special sequences of algorithms. Our present results could also be obtained by that method, which is not essentially different from the one used here.

§ 2. *Weights of algorithms.* Let $w(x)$ be a function defined on the rationals in $-1 < x < 1$. Then to any equation $a_i = a_{i+1}q_i + r_i$ $(i = 1, \ldots, n;\ r_i = \varepsilon_i a_{i+2}\ (i < n),\ r_n = 0)$ we assign as its „weight" the value $w(r_i/a_{i+1})$. The weight of an algorithm will be defined as the sum of the weights of its equations. The function $w(x)$ will be

called a *proper weight function* if the weight of any algorithm only depends on a_1 and a_2, that is to say, if the sum of the weights of any algorithm is an invariant.

To any set S of the type described in § 1 we can make correspond a proper weight function such that the invariant induced by it equals the length of the S-algorithm.

Let $L_S(a_2/a_1)$ denote the length of the S-algorithm for a_1, a_2, which actually only depends on the ratio of a_1 and a_2. Put $L_S(0)=0$. Consider the equation

$$a = bq + r \qquad (a > b > |r| \geq 0) \tag{2.1}$$

Naturally, $L_S(b/a)$ only depends on b and on the residue class of a mod b:

$$L_S(b/a_1) = L_S(b/a_2) \qquad (a_1 > a_2 > b > 0,\ a_1 \equiv a_2 \pmod{b}). \tag{2.2}$$

Therefore, the function $w_S(x)$ is uniquely defined for all rationals in $-1 < x < 1$ by

$$w_S(r/b) = L_S(b/a) - L_S(|r|/b) \tag{2.3}$$

Now (2.3) is the weight assigned to equation (2.1), and it is clear that the weight of any algorithm (1.1) equals $L_S(a_2/a_1)$. This is, for S fixed, an invariant, and hence $w_S(x)$ is a proper weight function. And, if $w_S(x)$ is known, the length of the S-algorithm for a_1, a_2 can be calculated from any other algorithm for the same pair.

The difficulty lies, of course, in determining $w_S(x)$ if S is given. A few things can be said in general. First

$$w_S(x) = 1 \text{ if } x \in S, \tag{2.4}$$

which follows from (2.3) and from

$$L_S(b/a) = 1 + L_S(|r|/b) \quad (a>b>|r|\geq 0,\ r \equiv a \pmod{b},\ r/b \in S) \tag{2.5}$$

This formula arises from the remark that the S-algorithm for b, $|r|$ is obtained from the S-algorithm for a, b by cancelling the first equation.

In the next section we shall specialize S to S_1, S_2, S_3, S_4. For typographical reasons, we shall write L_i and w_i instead of L_{S_i}, w_{S_i}.

More general invariants can be obtained by replacing $L_S(b/a)$ by any expression $M_S(a, b)$, depending uniquely on the S-algorithm for a and b. For instance: $M_S(a_1, a_2) = q_1 + q_2 + \ldots + q_n$, where the q's are the quotients occurring in the S-algorithm. In some cases the weight functions corresponding to such invariants can be evaluated explicitly (see the example at the end of § 5).

§ 3. *The Euclidean Algorithm.* Let S_1 be the set $0 \leq x < 1$, then the S-algorithm is the euclidean algorithm. L_1 and w_1 are length and weight corresponding to this choice of S.

T h e o r e m 1. We have

$$w_1(x) = 1 \text{ if } 0 \leq x < 1 \qquad \text{and if } x = -\tfrac{1}{2}, \tag{3.1}$$

$$w_1(x) = 2 \text{ if } -\tfrac{1}{2} < x < 0, \tag{3.2}$$

$$w_1(x) = 0 \text{ if } -1 < x < -\tfrac{1}{2}. \tag{3.3}$$

P r o o f. The case $0 \leq x < 1$ follows from (2.4). If $x = -\frac{1}{2}$ we have, by (2.3), $w_1(-\frac{1}{2}) = L_1(^2/_3) - L_1(^1/_2) = 2 - 1 = 1$. This proves (3.1).

Now assume $a = bq + r$, $a > b > 0$, $-^1/_2 b < r < 0$. Then we have $b + r \equiv a \pmod{b}$, $(b + r)/b \in S_1$; hence, by (2.5)

$$L_1(b/a) = 1 + L_1((b + r)/b). \tag{3.4}$$

We next apply (2.5) once again, remarking that $b > b + r > |-r| > 0$, $-r \equiv b \pmod{b + r}$, $-r/(b + r) \in S_1$. This leads to

$$L_1((b + r)/b) = 1 + L_1(-r/(b + r)).$$

Finally we have, by (2.2),

$$L_1(-r/(b + r)) = L_1(-r/b)$$

Now (2.3) shows that

$$w_1(r/b) = L_1(b/a) - L_1(-r/b) = 2.$$

Next assume $-b < r < -^1/_2 b$. Then (3.4) is still valid. Further, by (2.5), we have

$$L_1(-r/b) = 1 + L_1((b + r)/|r|), \tag{3.5}$$

as $b > -r > b + r > 0$, $b + r \equiv b \pmod{r}$, $(b + r)/(-r) \in S$. Finally, by (2.2),

$$L_1((b + r)/b) = L_1((b + r)/|r|), \tag{3.6}$$

as $b > -r > b + r > 0$, $b \equiv -r \pmod{b + r}$.

Now (3.4), (3.5), (3.6) and (2.3) prove (3.3).

E x a m p l e. Consider the following algorithm for the pair 77, 53: $77 \equiv -29$ (53), $53 \equiv -5$ (29), $29 \equiv 4(5)$, $5 \equiv -3$ (4), $4 \equiv 1$ (3), $3 \equiv 0$ (1). We have $w_1(-29/53) = 0$, $w_1(-5/29) = 2$, $w_1(4/5) = 1$, $w_1(-3/4) = 0$, $w_1(1/3) = 1$, $w_1(0) = 1$. Hence the length of the euclidean algorithm for a_1, a_2 equals

$$0 + 2 + 1 + 0 + 1 + 1 = 5.$$

If we consider an L.R.A., then the weight of each equation is 1 or 2 according to $\varepsilon = +1$ of $\varepsilon = -1$. Therefore the result of GOODMAN and ZARING, mentioned in § 1, is a special case of our theorem.

§ 4. *The Least Remainder Algorithm.* This is the S_2-algorithm, where S_2 is given by $-\frac{1}{2} < x \leq \frac{1}{2}$. Let θ be defined by

$$\theta = -\tfrac{1}{2} + \tfrac{1}{2}\sqrt{5} = [0,\ 1,\ 1,\ 1,\ \ldots] = 0.61803399\ldots;$$

here the notation $[q_0, q_1, q_2, \ldots]$ is the usual one for the continued fraction $q_0 + 1/\{q_1 + 1/(q_2 + \ldots$.

T h e o r e m 2. We have $w_2(x) = \psi(x)$, where $\psi(x)$ is defined by

$$\psi(x) = 1 \ (|\, x\,| < \theta),\ \psi(x) = 0 \ (\theta < |\, x\,| < 1)$$

P r o o f. According to (2.3) we have to show that

$$L_2(b/a) = \psi(r/b) + L_2(|\, r\,|/b) \tag{4.1}$$

for all integers a, b, r satisfying

$$a > b > |\, r\,|,\ a \equiv r \pmod{b}. \tag{4.2}$$

First we remark that it is sufficient to prove the theorem for $x > 0$. For, $L_2(b/a)$ only depends on the absolute value of the residue of a mod b; hence, by (2.3), $w_2(x) = w_2(-x)$. Next we note that we may restrict ourselves to $\frac{1}{2} < x < 1$, as for $0 \leq x \leq \frac{1}{2}$ we can apply (2.4).

We shall deal with the general case of (4.1) by induction with respect to b. For $b = 1$ it is true, as $L_2(1/a) = 1$, $L_2(0) = 0$. Next we shall prove (4.1) under the assumption that the analogous relation has been established for all values of b' not exceeding b. We may and do assume that $\frac{1}{2}b < r < b$.

We have, by the assumption of induction,

$$L_2(r/b) = \psi((b-r)/r) + L_2((b-r)/r), \tag{4.3}$$

since $b > r > |\, b - r\,|$, $b \equiv b - r \pmod{r}$. Further, by (2.5),

$$L_2(b/a) = 1 + L_2((b-r)/b), \tag{4.4}$$

since $a > b > |\, r - b\,|$, $r - b \equiv a \pmod{b}$, $(r-b)/b \in S_2$.

Finally, by (2.2),

$$L_2((b-r)/b) = L_2((b-r)/r) \tag{4.5}$$

In order to prove (4.1) from (4.3), (4.4) and (4.5) it remains to show that

$$\psi(r/b) + \psi((b-r)/r) = 1.$$

This is clear; if x is a positive rational, $0 < x < 1$, then the numbers x and $x^{-1} - 1$ are separated by θ. The proof is now complete.

Corrolaries. From $w_2(x) \leq 1$ $(-1 < x < 1)$ it follows that no algorithm is shorter than the L.R.A. Furthermore it follows that an algorithm has the same length as the L.R.A. for the same pair, if, and only if, all its equations have weight 1, that is, if $a_{i+2} < \theta a_{i+1}$ $(i = 1, \ldots, n-1$; the notation being the one of (1.1)) A numerical example:

$$29 \equiv 11(18), \quad 18 \equiv -3(11), \quad 11 \equiv -1(3), \quad 3 \equiv 0(1),$$
$$29 \equiv -7(18), \quad 18 \equiv -3(7), \quad 7 \equiv 1(3), \quad 3 \equiv 0(1);$$

the second one is the L.R.A. Indeed we have $11/18 < \theta$, $3/11 < \theta$, $1/3 < \theta$.

§ 5. *The Greatest Remainder Algorithm.* This is the S_3-algorithm, where S_3 consists of the intervals $-1 < x \leq -\frac{1}{2}$, $\frac{1}{2} < x < 1$ and of the number 0.

Theorem 3. We have $w_3(0) = 1$, and

$$w_3(x) = n\left(\frac{1}{n+1} \leq |x| < \frac{1}{n},\ n = 1, 2, 3, \ldots\right).$$

Proof. By (2.4) we may restrict ourselves to $0 < |x| < \frac{1}{2}$ (the case $x = \frac{1}{2}$ is easily disposed of). In the general case we have to show that

$$L_3(b/a) - L_3(|r|/b) = n \tag{5.1}$$

if

$$a > b > |r| > 0,\ a \equiv r \pmod{b},\ n|r| < b \leq (n+1)|r|. \tag{5.2}$$

This is true if $n = 1$, for then we have $|r| \geq \frac{1}{2}b$. We next take $n > 1$, and we apply induction with respect to n. As (5.2) implies $b > b - |r| > |r| > 0$, $b \equiv |r| \pmod{b - |r|}$, $(n-1)|r| < b - |r| \leq n|r|$, we may assume

$$L_3((b - |r|)/b) - L_3(|r|/(b - |r|)) = n - 1. \tag{5.3}$$

We have, by (2.5) and by (2.2), respectively,

$$L_3(b/a) = 1 + L_3((b - |r|)/b),\ L_3(|r|/b) = L_3(|r|/(b - |r|)). \tag{5.4}$$

Now (5.1) follows from (5.3) and (5.4).

Corrolaries. 1. As $w_3(x) \geq 1$ for all $x(-1 < x < 1)$, we infer that no algorithm is longer than the G.R.A. for the same pair. If an algorithm for a, b has the same length as the G.R.A., then

each equation has weight 1. This means that the algorithm exactly follows the G.R.A., only with a possible difference in the equation before the last one, where a value of $x = +\frac{1}{2}$ may occur.

2. The length of the G.R.A. is easily expressed in terms of the E.A. by calculating the weight of the latter. The result is, if

$$a/b = [q_0, q_1, q_2, \ldots, q_n],$$

that

$$L_3(b/a) = q_1 + q_2 + \ldots + q_n, \text{ unless } a \equiv 0 \pmod{b}.$$

3. More generally we have: if an algorithm for a_1, a_2 is given by (1.1), then

$$\left[\frac{a_1}{a_2}\right] + \left[\frac{a_2}{a_3}\right] + \ldots + \left[\frac{a_n}{a_{n+1}}\right] \tag{5.5}$$

is an invariant. For, if $\{x\}$ denotes the greatest integer $< x$, then (5.5) equals

$$\left\{\frac{a_1}{a_2}\right\} + \left\{\frac{a_2}{a_3}\right\} + \ldots + \left\{\frac{a_n}{a_{n+1}}\right\} + 1 =$$

$$= \left\{\frac{a_1}{a_2}\right\} + w_3(a_3/a_2) + \ldots + w_3(a_{n+1}/a_n) + 1,$$

and $w_3(0/a_{n+1}) = 1$. Hence (5.5) equals $L_3(a_2/a_1) + \{a_1/a_2\}$.

A special result was proved in [1] Theorem 3 which, combined with [1] Theorem 1, leads to the assertion that the expression (5.5) has the same value for the E.A. as for the L.R.A.

§ 6. *The Anti Euclidean Algorithm.* This is the S_4-algorithm, where S_4 is the set $-1 < x \leq 0$. It is remarkable that $w_4(x)$ is much more complicated than w_1, w_2, w_3.

We define the functions $\chi(x)$ and $\varphi(x)$ as follows. Let the rational number $x (0 < x < 1)$ be given by its continued fraction

$$x = [0, q_1, q_2, \ldots, q_n] \quad (q_n > 1) \tag{6.1}$$

We define q_{n+1}, q_{n+2}, ... by $q_{n+1} = 1$, $q_{n+2} = q_{n+3} = \ldots = 0$ and we put

$$\chi(x) = q_1 + q_3 + q_5 + \ldots, \quad \varphi(x) = q_2 + q_4 + q_6 + \ldots$$

Theorem 4. $L_4(x) = \varphi(x) \quad (0 < x < 1)$.

The proof is of the same type as the proofs of the previous theorems. We only give a brief sketch: from (2.5) we infer, if x is

given by (6.1), that $L_4(x) = 1 + L_4(y)$, where $y = 1 - [0, q_2, q_3, \ldots, q_n]$. Now the identity

$$[0, 1, p_1, \ldots, p_n] + [0, p_1 + 1, p_2, \ldots, p_n] = 1$$

enables us to carry out an induction step.

An immediate consequence of (2.4) and of Theorem 4 is

Theorem 5.

$$w_4(x) = 1 \qquad (-1 < x \leq 0)$$
$$w_4(x) = \chi(x) - \varphi(x) \qquad (0 < x < 1)$$

University of Amsterdam
University of Kentucky

(Received 4.4.1953)

REFERENCES

[1] A. W. Goodman and W. M. Zaring, Euclid's Algorithm and the Least Remainder Algorithm. Amer. Math. Monthly, *59*, 156—159, (1952).

[2] Uspensky and Heaslet, Elementary Number Theory. New York 1939, pp. 26—28 and pp. 45—51.