Jebelean-Weber's Algorithm without Spurious Factors

Sidi Mohamed Sedjelmaci

LIPN, CNRS UPRES-A 7030
Université Paris-Nord,
Av. J.B.-Clément, 93430 Villetaneuse, France.
sms@lipn.univ-paris13.fr

Abstract

Tudor Jebelean and Ken Weber introduced in [10, 4] an algorithm for finding a (a, b)-pairs satisfying $au + bv \equiv 0 \pmod{k}$, with $0 < |a|,\ |b| < \sqrt{k}$. It is based on Sorenson's "k-ary reduction" algorithm ([9]). One of the major drawbacks is that this algorithm do not preserves the GCD's. We present a modified version which avoids this problem.

Keywords: Integer greatest common divisor (GCD); Extended GCD algorithm; Complexity analysis; Number theory.

1 Introduction

Given two integers a and b, the greatest commun divisor of a and b, denoted $GCD(a, b)$, is the largest integer which divides both a and b. Applications for GCD algorithms include computer arithmetic, integer factoring, cryptology and symbolic computation [6, 8, 5] . In [9], Sorenson proposed the "right-shift k-ary algorithm". It is based on the following reduction. Given two positive integers $u > v$ relatively prime to k (i.e., (u, k) and (v, k) are coprime), pairs of integers (a, b) can be found that satisfy

$$\begin{aligned} au + bv \ &\equiv\ 0 \pmod{k}, \\ \text{with} \quad &0 < |a|,\ |b| < \sqrt{k}. \end{aligned} \tag{1}$$

If we perform the transformation (also called "k-ary reduction"):

$$(u, v) \longmapsto (u', v') = (|au + bv|/k, \min(u, v)),$$

which replaces u with $u' = |au + bv|/k$, the size of u is reduced by roughly $\frac{1}{2}\log_2(k)$ bits since

$$|au + bv|/k \ \leq 2\max(|a|, |b|)\ \frac{u}{k}\ < \frac{2u}{\sqrt{k}}. \tag{2}$$

Sorensen suggests table lookup to find sufficiently small a and b satisfying (1). By contrast, Jebelean [4] and Weber [10] both propose an easy algorithm, which finds such small a and b that satisfy (1) with time complexity $O(n^2)$. This latter algorithm we call the "Jebelean-Weber algorithm", or *JWA* for short. A GCD algorithm based on this reduction works very well in practice and is included in Gnu MP multiprecision library [2]. However this GCD algorithm does not preserves the GCD since:

$$GCD(v, |au + bv|/k) = \alpha\ GCD(u, v), \quad \text{with } \alpha | a,$$

whence some spurious factors must be eliminated (see example in Section 4). Although its good time behaviour, this drawbacks affects somehow the efficiency of the GCD algorithm. In this present work we show how a slightly modified version of JWA easily ovoids this problem. Not only this modified version is desirable for GCD computations but it is also needed in many other applications such as Jacobi symbol computation or modular inverse to mention only a few [6].

The paper is organized as follows. Notations and definitions are given in Section 2. In Section 3, we recall the Jebelean-Weber algorithm and propose a modified version. Section 4 deals with the correctness of our algorithm and we conclude with some remarks in Section 5.

2 Notation

Throughout this paper, we restrict ourselves to the set of non-negative integers. Let u and v be two such (non-negative) integers, u and v are respectively n-bits and p-bits numbers with $u \geq v \geq 1$. Let k be an integer parameter s.t. $k = 2^m$ with $m \geq 2$.

Given a non-negative integer $x \in N$, $\ell_2(x)$ represents the number of significant bits of an non-negative integer x, not counting leading zeros: $\ell_2(x) = \lfloor \log_2(x) \rfloor + 1$, if $x \geq 1$ and $\ell_2(0) = 0$. So $n = \ell_2(u)$, $p = \ell_2(v)$ and p satisfies $2^{p-1} \leq v < 2^p$. We let $\rho = \rho(u, v) = \ell_2(u) - \ell_2(v) + 1$. Thus, we obtain $2^{\rho-2} < u/v < 2^\rho$. We assume that $p > 2m + \rho + 1$.

Let a, b and c be positive integers, the integer $x = a \bmod b$ is the unique non-negative integer x such that

$$0 \leq x \leq b - 1 \quad \text{and} \quad a - x = 0 \bmod b.$$

Note that this notation still holds when $a < 0$.

If b is relatively prime to k, then $r = a/b \bmod k$ is the unique non-negative integer r such that

$$0 \leq r \leq k - 1 \textit{ and } \ r\ b \equiv a \pmod{k}.$$

As noticed by many authors the main difficulty in GCD algorithms happens when the input data u and v are roughly of the same size [9, 4, 10]. So we

shall assume that when Sorenson's reduction is applied : $n - p \leq m/2 - 1$ (or $\rho \leq m/2$) so that $u/v < 2^{m/2} = \sqrt{k}$. Otherwise, we usually apply a more efficient reduction: the *bmod*, defined as:

$$bmod(u, v) = |u - (u/v \mod 2^{\rho})v|/2^{\rho}.$$

Recall that the determinant of a matrix $M = \begin{pmatrix} c & d \\ a & b \end{pmatrix}$ is $\det M = cb - ad$. The extended version of Euclid GCD algorithm is noted *EEA* [5]. It is tightly linked with the continued fractions [3, 5] and is important for its multiple applications in cryptology and computer algebra.

3 The Algorithms

3.1 The Jebelean-Weber Algorithm: `JWA`

Let us first recall the JWA as stated in [10].

Input: $x,\ y > 0,\ k > 1$, and $\gcd(k, x) = \gcd(k, y) = 1$.
Output: (n, d) such that $0 < n,\ |d| < \sqrt{k}$, and $ny \equiv dx \pmod{k}$.

$r := x/y \bmod k$; /* initialization */
$f_1 = (n_1, d_1) := (k, 0)$;
$f_2 = (n_2, d_2) := (r, 1)$
  while $n_2 \geq \sqrt{k}$ **do**
    $f_1 := f_1 - \lfloor n_1/n_2 \rfloor f_2$
    swap (f_1, f_2)
  endwhile
return f_2

Fig. 1. *The Jebelean-Weber Algorithm:* `JWA`

When (n, d) is the output result of JWA, the pair $(a, b) = (d, -n)$ (or $(-d, n)$) satisfies the property $au + bv = 0 \bmod k$. The algorithm `JWA` is nothing but the extended version of Euclid `EEA` applied to the pair $(k, u/v \bmod k)$, where only one column is added instead of two for `EEA` (see [5]), and they only differ on their exit test.

3.2 The Modified Jebelean-Weber Algorithm: M-JWA

We give below a modified version that ovoids spurious factors introduced in JWA.

Input: $x,\ y > 0,\ k \geq 4$ such that $\gcd(k, x) = \gcd(k, y) = 1$ and $x/y < \sqrt{k}$.
Output: A 2×2 integer matrix $M = \begin{pmatrix} n_1 & d_1 \\ n_2 & d_2 \end{pmatrix}$ such that

$$0 < n_2,\ |d_2| < \sqrt{k},\ \ n_2 y \equiv d_2 x \pmod{k} \ \text{ and } \ n_1 y \equiv d_1 x \pmod{k}\ .$$

$r := x/y \bmod k$; /* initialization */
$f_1 = (n_1, d_1) := (k, 0)$;
$f_2 = (n_2, d_2) := (r, 1)$;
while $n_2 \geq \sqrt{k}$ **do**
$f_1 := f_1 - \lfloor n_1/n_2 \rfloor f_2$
swap (f_1, f_2)
endwhile
return $M = \begin{pmatrix} n_1 & d_1 \\ n_2 & d_2 \end{pmatrix}$

Fig. 1. *The Modified Jebelean-Weber Algorithm:* M-JWA

The new transformation assiociated with the output matrix of M-JWA is defined by $(u, v) \leftarrow (R_1, R_2)$ with:

$$R_1 = |n_1 v - d_1 u|/k \text{ and } R_2 = |n_2 v - d_2 u|/k. \tag{3}$$

We will prove in the next Section that this transformation $(u, v) \leftarrow (R_1, R_2)$ preserves the GCD, i.e.: $GCD(R_1, R_2) = GCD(u, v)$ and avoids the spurious factors of algorithm JWA.

4 Correctness

Before proving that indeed, M-JWA preserves the GCD, we first recall below some well known properties [3, 5] of EEA that are also valid for JWA as well as for M-JWA. Let $(n_s, d_s)_{s \geq 1}$ be the pair of sequences corresponding to the successive results of f_2 in JWA or M-JWA, then $\forall s \geq 1$ we have

- $n_s > 0$ and $d_s d_{s+1} < 0$
- $n_s/d_s \equiv x/y \pmod{k}$
- $n_s d_{s+1} - n_{s+1} d_s = \pm k$
- $(n_s)_s$ is decreasing and $(d_s)_s$ is increasing.

Lemma 4.1 *The ouptout of JWA satisfies* $n_2 y - d_2 x \equiv 0 \pmod{k}$ *and* $0 < n_2, |d_2| < \sqrt{k}$.

Proof : ([10]) In the last iteration i of `JWA` or `M-JWA` n_i must meet the condition $n_i < \sqrt{k} < n_{i-1}$. Hence, since $n_{i-1}|d_i| + n_i|d_{i-1}| = k$, $n_{i-1}|d_i| < k$ and $|d_i| < k/n_{i-1} \leq \sqrt{k}$. *QED*

We prove in the following that the output integer matrix of `M-JWA` enjoys more interesting properties.

Lemma 4.2 *Let $u \geq v \geq 1$ and $k = 2^t \geq 4$ be three positive integers such that $u/v < \sqrt{k}$. Let $\begin{pmatrix} c & d \\ a & b \end{pmatrix}$ be the output integer matrix of* `M-JWA`*, then $G = (|du - cv|)/k$ is a positive integer such that $0 \leq G \leq v$.*

Proof : First, G is an integer since $c/d \equiv a/b \equiv u/v \bmod k = r$. Moreover, if $r < \sqrt{k}$ then $c = k$, $d = 0$ and $G = v$. Otherwise $k > c \geq \sqrt{k}$ and since $|d| \leq |b|$, we proceed in two cases:
Case 1: If $|b| = |d|$, then this case happens only when $b = -1$, $d = 1$ and $\lfloor k/c \rfloor = 1$. Since $k > c > \sqrt{k} > u/v$, we obtain

$$G = |u - cv|/k = |u/v - c|(v/k) = (c - u/v)(v/k) < (c/k)v < v.$$

Case 2: If $|b| > |d|$. We have $G \leq (|d|u + cv)/k \leq (\frac{|d|u}{kv} + c/k)v$. Let us prove that $\frac{|d|u}{kv} + c/k < 1$ or $u/v < \frac{k-c}{|d|}$. From $|b| > |d|$ we obtain $\frac{|b|-1}{|d|} \geq 1$ and since $k = c|b| + a|d|$, we obtain the result

$$\frac{k-c}{|d|} = \frac{c|b| + a|d| - c}{|d|} = c(\frac{|b|-1}{|d|}) + a \geq \sqrt{k} > u/v.$$

QED

Lemma 4.3 *Let $u \geq v \geq 1$ and $k \geq 1$ be three integers such that $GCD(u, k) = GCD(u, k) = 1$. Let $M = \begin{pmatrix} c & d \\ a & b \end{pmatrix}$ be an integer matrix with $\det M = |cb - ad| = k$. If there exist two integers R_1, R_2 satisfying*

$$k \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = M \begin{pmatrix} u \\ v \end{pmatrix}, \text{ then } GCD(R_1, R_2) = GCD(u, v).$$

Proof : Let $\alpha = GCD(u, v)$ and $\beta = GCD(R_1, R_2)$ then $k\ R_1 = (cu + dv)$ so $\alpha | kR_1$ but $GCD(u, k) = 1$ then $GCD(\alpha, k) = 1$ and $\alpha | R_1$. Similarly $k\ R_2 = (au + bv)$ and $\alpha | R_2$. Hence $\alpha | \beta$. Moreover, since $|cb - ad| = k \neq 0$, M^{-1} exists and $\begin{pmatrix} u \\ v \end{pmatrix} = k \times M^{-1} \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = k \times \epsilon/k \times \begin{pmatrix} bR_1 - dR_2 \\ -aR_1 + cR_2 \end{pmatrix}$, where $\epsilon = \pm 1$, hence $\beta | \alpha$ and the result $\alpha = \beta$. *QED*

Remark: It is worth to note that this Lemma generalizes a well known result in the case $k = \pm 1$, i.e.: if $\begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = M \begin{pmatrix} u \\ v \end{pmatrix}$, and $\det M = \pm 1$ then $GCD(R_1, R_2) = GCD(u, v)$. This situation occurs in EEA.

Proposition 4.1 *Let* $\begin{pmatrix} n_1 & d_1 \\ n_2 & d_2 \end{pmatrix}$ *be the output integer matrix of* `M-JWA`. *If* $\begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \begin{pmatrix} |n_1 v - d_1 u|/k \\ |n_2 v - d_2 u|/k \end{pmatrix}$, *then* R_1 *and* R_2 *are two integers satisfying* $0 \leq R_1 \leq v$, $0 \leq R_2 \leq 2u/\sqrt{k}$ *and* $GCD(R_1, R_2) = GCD(u, v)$.

Proof : We have $k \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = N \begin{pmatrix} u \\ v \end{pmatrix}$, where N is one of the four following matrices $N_1 = \begin{pmatrix} -d_1 & n_1 \\ -d_2 & n_2 \end{pmatrix}$, $N_2 = \begin{pmatrix} d_1 & -n_1 \\ d_2 & -n_2 \end{pmatrix}$, $N_3 = \begin{pmatrix} -d_1 & n_1 \\ d_2 & -n_2 \end{pmatrix}$ or $N_4 = \begin{pmatrix} d_1 & -n_1 \\ -d_2 & n_2 \end{pmatrix}$. Then the result derives straightforwards from Lemma 4.1, 4.2, 4.3 and relation (2).

QED

Example: If $(u, v) = (28865, 19203)$ and $k = 2^6 = 64$. Note that $GCD(u, v) = 1$. We obtain in turn $u/v \bmod k = 1/3 \bmod 64 = 43$, and $M = \begin{pmatrix} 21 & -1 \\ 1 & 3 \end{pmatrix}$. Hence

$$R_1 = |u + 21v|/64 = 6752, \qquad R_2 = |3u - v|/64 = 1053.$$

The `JWA` algorithm uses the transformation $(u, v) \leftarrow (v, R_2)$. However $GCD(v, R_2) = GCD(19203, 1053) = 3 \neq GCD(u, v)$, while, with `M-JWA` algorithm, we obtain $GCD(R_1, R_2) = GCD(6752, 1053) = GCD(u, v) = 1$. The spurious factor 3 has been eliminated.

5 Conclusion

In this paper, we have showed that a slight modification easily avoids the spurious factors introduced by the early version of `JWA`. Basically the spurious factors are eliminated without extra amount of time. As a matter of fact, the `JWA` algorithm yet provides the matrix $M = (f_1, f_2)$ which is also output of `M-JWA`, but it only uses the vector f_2 as a GCD transformation. By contrast, in `M-JWA` algorithm, the full matrix $M = (f_1, f_2)$ is also used to avoid spurious factors, in a same time. Although the algorithm `JWA` works well in practice, we should expect a better running time with our improved version `M-JWA`.

References

[1] **A.V. Aho, J.E. Hopcroft and J.D. Ullman.** *The Design and Analysis of Computer Algorithms*, Addison Wesley, 1974.

[2] **Gnu MP 4.1.2.** online reference manuel, *http://swox.com/gmp/manuel*, /index.html. 2002.

[3] **G.H. Hardy and E.V. Wright.** *An Introduction To The Theory Of Numbers*, Oxford University Press., London, 1979.

[4] **T. Jebelean.** A Generalization of the Binary GCD Algorithm, *in Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'93)*, 1993, 111-116.

[5] **D.E. Knuth.** *The Art of Computer Programming*, Vol. 1-2, 2nd ed., Addison Wesley, 1981.

[6] **A.J. Manezes, P.C.van Oorschot, S.,A., Vanstone.** *Handbook of Applied Cryptography*, Vol. 1-2, 2nd ed., CRC Press, 1997.

[7] **A. Schönhage.** Schnelle Berechnung von Kettenbruchentwicklugen, *Acta Informatica*, 1, 1971, 139-144.

[8] **J. von zur Gathen, J. Gerhard.** *Modern Computer Algebra*, 1st ed., Cambrige University Press, 1999.

[9] **J. Sorenson.** Two Fast GCD Algorithms, *J. of Algorithms*, 16, 1994, 110-144.

[10] **K. Weber.** Parallel implementation of the accelerated integer GCD algorithm, *J. of symbolic Computation (Special Issue on Parallel Symbolic Computation)*, 21, 1996, 457-466.