On acceleration of the $k$-ary GCD Algorithm

Ismail Amer
Institute of Computational Mathematics and Informational Technologies, Kazan Federal University, Kremlevskaya 18, Kazan, 410008, Russia
E-mail: safadi121979@yahoo.com

Abstract. In our paper we analyze a method of acceleration of the $k$-ary Algorithm of finding the greatest common divisor for long natural numbers. The $k$-ary GCD algorithm was invented in 1990 by J. Sorenson. We show that a small modification of Sorenson’s Algorithm gives a sufficient acceleration of its work. This is useful for applications using the GCD operation like calculations in finite fields, generation of Cryptography keys etc.

1. Introduction

The $k$-ary GCD algorithm (briefly, KARY) was invented by J. Sorenson [1], [2]. Jebelean and Weber independently improved it in [3], [4] by adding a new operation $dmod$, which is applied when $A$ is sufficiently larger than $B$.

Let $k > 1$ be any natural and $A \geq B > 0$ be integers that have no common divisors with $k$. The main idea of a step of the algorithm is to find small integers $x$ and $y$ such that

$$Ax + By \equiv 0 \mod k,$$

then compute $C = |Ax + By|/k$ and replace the pair $(A,B)$ by a minor pair $(B,C)$ or $(C,B)$ depending on whether $B > C$ holds. Additionally it may require a reduction of $C$ by $gcd(C,k)$. The algorithm is based on the following theorem:

Theorem 1: (J. Sorenson). For any $A \geq B > 0$ incomparable with $k$ there exist non-zero, $y, |x|, |y| \leq m, m = \lceil \sqrt{k} \rceil$, such that relation

$$Ax + By \equiv 0 \mod k$$

holds.

Proof. Fix $A$ and $B$ such as in the theorem and define set $M = \{x,y\}$ to consist of all pairs $(x,y)$ satisfying $-[\lfloor m/2 \rfloor] < x, y < \lfloor m/2 \rfloor, x \neq 0, y \neq 0$.

Note that the power of $M$ is greater or equal to $k$. Define a function $h$ realizing a map from $M$ to $[0; k-1]$ as follows:

$$h(x,y) = (Ax + By) \mod k.$$

Let consider two possible cases:

- Function $h$ is injective. Then $h$ performs a 1-1 map from $M$ to $Z_k$ and there exist non-zero $x, y$ with $h(x,y) = 0$. Clearly, $(x, y)$ satisfies the theorem. In this case $|x|, |y| \leq \lfloor m/2 \rfloor$. 


• Function $h$ is not injective. Then there exist different pairs $(x_1, y_1)$ and $(x_2, y_2)$ such that $h(x_1, y_1) = h(x_2, y_2)$. Define $x = x_1 - x_2$ and $y = y_1 - y_2$, then

$$Ax + By \mod k = ((Ax_1 + By_1) - (Ax_2 + By_2)) \mod k = 0$$

This proves the theorem.

Remark. The KARY has a minor disadvantage that the GCD of pair $(B; C = (Ax + By) \mod k)$ is not obligatory to be equal to the origin GCD of $(A, B)$. But $gcd(A, B)$ is a factor of $gcd(B, C)$. So the final value of GCD in $k$-ary method has the origin GCD as a factor. To find the origin GCD $d$ we need to add at the end a final calculation:

$$d' = gcd(B, d''), d = gcd(A, d'),$$

where $d''$ is the gcd obtained by the $k$-ary algorithm and $d, d'$ are calculated with the classical Euclidian GCD Algorithm.

2. Analysis of the $k$-ary GCD Algorithm

The effectiveness of the KARY depends on the ratio $A/C$ reached at steps of the algorithm. By theorem 1 it exceeds $\sqrt{k}/2$. Indeed,

$$\frac{A}{C} = \frac{kA}{xA + yB} \geq \frac{kA}{2\sqrt{k}A} = \frac{\sqrt{k}}{2}$$

Since, $k$ is usually chosen to be a power of $2 k = 2^l$ with $L$ equal to 16, 32 or even 64, then the KARY essentially overcomes the classical Euclidian Algorithm that has the ratio $A/C$ by a D.Knuth's remark not exceeding 3 in more than 70 percents iterations if we compare the number of iterations. But in general the KARY loses to the EA since the latter implements operation $C = A \mod B$ faster than the KARY computes its $C$.

Sorenson in [2] suggested several ways to speed up the implementation of the KARY. The simplest way is to use precomupted tables. Let (1) hold and $q = AB^{-1} \mod k$ then

$$Ax + By \equiv 0 \rightarrow y \equiv -qx \mod k \rightarrow y = -qx + ks \text{ for some } s \in Z.$$

Since choice of $x$ and $y$ in relation (1) completely depends on $q \in [1; k)$ then we can build a table function

$$f: [1; k) \rightarrow D = ((x, y)|1 \leq x \leq \sqrt{k}, |y| \leq \sqrt{k}, y \equiv -qx \mod k).$$

At a stage of the KARY we compute $q = AB^{-1} \mod k$ and choose the corresponding values of $x$ and $y$ from the table $f$. An additional speed-up can be reached if we pre-compute table of inverses by module $k$. Since $k$ is a power of 2 $k = 2^l$ then we do not need to keep the complete table of inverses but keep the table of inverses by module $2^{l'}$ for some $l' < L$ and compute inverses by a large $L$ using so called Henzel's Lifting. This algorithm is described in [6].

With such modification at large numbers the KARY begins to work faster than the classical EA.

3. An acceleration of the KARY

A. Shift of the interval of $y - \text{values}$.

We add an additional hint allowing the KARY to work faster. Let return to theorem 1. If we analyze possible decision $(x, y)$ of relation (1) we see that value $C = |(Ax + By)|/k$ is less when $y < 0$ since additives $Ax$ and $By$ have opposite signs and reduce one another. By theorem 1, case $y < 0$ occurs in average in a half of all computations. The our idea is to enlarge the portion of cases $y < 0$ in computations of $(x, y)$.

We note that by theorem 1 values of variable $x$ are restricted by set $[1; \sqrt{k}]$ while $y$ is varying in the interval $D = [-\sqrt{k}; \sqrt{k}]$. In fact, we can replace set $D$ by a set $D_t = [-\sqrt{k} - t; \sqrt{k} - t]$ for $t > 0$ with
remaining theorem 1 valid. Direct computations show the speed-up of the KARY when we force \( t \) to take subsequently values 0, 1, 2, … Notice that case \( t = 0 \) corresponds to the usual implementation of the KARY algorithm.

The question arises what value of parameter \( t \) is optimal for the KARY speed? The answer depends on length of sets \( A, B \) and value of parameter \( k \).

B. Analysis of values of \( x \) and \( y \) in relation (1).

As we see earlier, values of \( x \) and \( y \) in the standard KARY depends on values of \( q \in [1; k] \), and conversely. For each \( x \leq \sqrt{k} \) there is a list of possible \( q \) and \( y \) corresponding to \( x \). Lists corresponding to different \( x \) can be intersected. Indeed, if \( (x, y) = (1, 1) \) is a decision of (1) then pair \( (x, y) = (2, 2) \) is also a decision, so choice of pair \( (x, y) \) depends on the traversal way. We use the following procedure:

- Force values of \( x \) subsequently take values \( 1, 2, \ldots, \sqrt{k} \).
- For each \( x \) compute \( y_1 = -qx \mod k \) and \( y_2 = y_1 + k \). Notice that \( y_1 < 0 \) and \( y_2 > 0 \).
- Check, if \( |y_i| \leq \sqrt{k} \) for \( i = 1, 2 \). If it holds, take the found pair \( (x, y) \) as decision to (1) and stop computation.

Now we consider possible \( q \) and \( y \) corresponding to different \( x \). We assume \( k = 2^L \) for even \( L \) so \( m = \sqrt{k} = 2^{L/2} \) is integer.

1. \( x = 1 \). Then, \( y = -qx = -q \), or \( y = -q + k \). From \( |y| \leq m \) we obtain \( 1 \leq q \leq m \) or \( k - m \leq q < k \).
2. \( x = 2 \). Then, \( y = -2q \) or \( y = -2q + k \), and \( (k - m)/2 < q < (k + m)/2 \).
3. \( x = 3 \). Then, \( (k - m)/3 < q < (k + m)/3 \), or \( 2(k - m)/3 < q < 2(k + m)/3 \).

We illustrate this procedure at \( k = 16 \). When \( x = 1 \) \( q \) satisfies \( 1 \leq q \leq 4 \) or \( 12 \leq q < 16 \). If \( q \in [6; 10] \). At \( x = 3 \) \( q = 5 \) or \( q = 9 \). This finishes the procedure since all possible values of \( q \) are already found. So at \( k = 16 \) and chosen way of enumeration of \( x \) from 1 to \( m \) the last possible value \( x = 4 \) is not reached.

Assume now that we shifted set of possible values of \( y, D = [-m; m] \) to the left by 1. Then, value \( y = m \) is not possible. At \( k = 16 \) this case corresponds to \( m = 4 \) and \( q = 12 \). In the shifted case pair \( (x, y) \) takes value \( 3, -4 \). and choice \( C = (3A - 4B)/16 \) is sufficiently better than original \( C = (A + 4B)/16 \).

This remark shows benefits of the shifting operation. The last problem is to find an optimal \( t \), on which set \( D \) should be shifted. Clearly, this \( t \) depends on length of numbers \( A, B \) and value of \( k \). The best case for pair \( (x, y) \) is reached at pair \( (x, y) \) when \( Ax \approx -By \). So relation \( x/|y| \) is inversely proportional to \( \alpha = A/B \).

As we mentioned earlier, an average value of \( \alpha \) in the classical Euclid Algorithm is small and changes between 2 and 3. In the KARY it depends on \( k \) and exceeds \( m/2 \). In the Approximating k-ary Algorithm by A. Ishmukhametov [5] it is compatible with \( 2k/3 \). So, the best shifting number \( t \) depends on \( k \) and current value of \( \alpha = A/B \). It can be chosen from a precomputed table of pairs \( (x, y) \) in which each value \( q \) is connected with several possible pairs \( (x, y) \). The best pair can be chosen from the table after computation of \( \alpha = A/B \). Since only elder digits of \( \alpha \) play a significant role, the parameter \( \alpha \) can be calculated with a rough approximation using several elder digits of \( A \) and \( B \).

Simple statistical observations lead us to a conclusion that best results should be concentrated near the middle of interval \([1; k]\). This was confirmed by experimental results.

4. Experimental results

We took \( k = 64 \) and performed a hundred GCD computations for pairs \( (A, B) \) containing 1000 decimal digits for each shift \( t \) taking values from 0 to \( k \). Then we computed an average time of GCD computation for each \( t \) and collected all results in figure 1.
We see that the interval shifting used at $k = 64$ helps to decrease time of computation from 1264 ms to 1001 ms saving about 25 percents of full time. Optimal value is observed at the middle of interval $[1; 64)$ at $t = 30$.

In our experiments we performed a common shift of interval $[-\sqrt{k}; \sqrt{k}]$ for $y$ to the left relative to all stages of GCD computation. An individual shifting at each stage of GCD computation should give a greater effect. But this requires a deeper analysis.

Our results can be applied to improve various algorithms using the GCD operation. In [6] an application of the KARY to computation of module inverses is investigated.

**Acknowledgment**

The research was funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities, project № 1.12878.2018/12.1.

**References**

[1] Sorenson J 1990 *The k-ary GCD Algorithm* (Univ.Wisc-Madison Tecn.Report) pp 1-20
[2] Sorenson J 1994 Two fast GCD Algorithms *J.Alg.* 16 110-44
[3] Jebelean T A 1993 Generalization of the binary GCD algorithm *Proceedings of the 1993 international symposium on Symbolic and algebraic computation* 1993 Aug 1 pp 111-6
[4] Weber K 1995 The accelerated integer GCD algorithm *ACM Transacations of Math.Software* 21 (1) 1-12
[5] Ishmukhametov S 2016 An approximating k-ary GCD algorithm *Lobachevskii Journal of Mathematics* 37(6) 723-9
[6] Ishmukhametov S and Mubarakov B 2017 Calculation of Bezout’s coefficients for the k-ary algorithm of finding GCD *Rus.Math.* 61 26-33