On implementation of an approximating $k$-ary GCD algorithm

Ismail F. Amer$^1$ and Mohammed H. Najajra$^2$

$^1$Institute of Computational Mathematics and Informational Technologies, Kazan Federal University (Kazan, Russia),
$^2$Department of Management Information Systems, Al Istiqal University (Jericho, Palestine)

E-mail: safadi121979@yahoo.com, mnajajra@mail.ru

Abstract. In our paper, we investigate some implementations of an approximating $k$-ary GCD algorithm (AKA) for calculating the greatest common divisor of natural numbers for given $A$ and $B$. This task has a great importance in the development of parameters for cryptographic encryption methods.

1. Introduction

Asymmetric modern cryptographic systems work with long numbers, for which it is necessary to calculate the greatest common divisor GCD [1]. The most common GCD algorithm is the classical Euclidean algorithm (CEA). Other algorithms also known with the common purpose of reducing the complexity of computing GCD, and reducing iterations and its calculation time.

One of the most advanced algorithms is the $k$-ary GCD algorithm (briefly, Kary), developed in the 1990s by J. Sorenson [2]-[4]. Compared to the classical Euclidean algorithm, the $k$-ary GCD algorithm allows to reduce iterations and computational complexity with a suitable choice of the parameter $k$ and other parameters.

Theorem (J. Sorenson) [3]. For any natural numbers $A, B$ and parameter $k > 1$, relatively prime with $A$ and $B$, there are integer numbers $x$ and $y$, satisfying identity:

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

Subsequently, several options for improving the $k$-ary GCD algorithm were proposed [5]-[7]. One of them is an approximating $k$-ary GCD algorithm developed in 2016 by Sh.T. Ishmukhametov [8]. The main difference from the Sorenson algorithm is that when switching from a pair $(A, B)$ to a pair $(B, C)$ having the same GCD as $GCD(A, B)$, where $C = \left\lfloor \frac{Ax + By}{k} \right\rfloor$. We select the $x, y$ values not by brute force, but look for such coefficient values so that $Ax + By$ is minimal.

2. Analysis of an approximating $k$-ary GCD algorithm

In this paper, we present an implementation of an approximating $k$-ary GCD algorithm and a comparison of its effectiveness with the mentioned GCD algorithms.
Consider, by example, the calculation of GCD using an approximating $k$-ary GCD algorithm. Let the numbers $A = 1485$, $B = 793$ be given. We perform one iteration of an approximating $k$-ary GCD algorithm for $k = 16$ (we choose the parameter $k$ equal to the power of two $k = 2^5$).

1. Find $a = A \mod k = 1485 \mod 16 = 13$, $b = B \mod k = 793 \mod 16 = 9$.
2. Find $q = AB^{-1} \mod k = 13 \cdot 9^{-1} \mod 16 = 13 \cdot 9 \mod 16 = 117 \mod 16 = 5$.
3. From the equation $Ax + By \equiv 0 \mod k$ we obtain
   
   \begin{align*}
   By & \equiv -Ax \mod k, \quad \text{or} \quad y \equiv -AB^{-1}x \mod k = -qx \mod k, \\
   y & \equiv -5x \mod 16, \quad \text{or} \quad y = -5x + 16s, s \in \mathbb{Z}.
   \end{align*}

4. At each iteration of the $k$-ary algorithm, the following set is determined

   \[ C = \frac{Ax + By}{k} \]

   We select integer coefficients $x$ and $y$ so that the value of $C$ is the smallest. To do this, we choose $x$ from the set $F_k = \{1,2,3,\ldots,k-1\}$, and $y$ we find from the condition $Ax \approx -By$, or $y \approx -rx$, where $r = A/B$ – rational number.

   Since $r = A/B$, it’s enough for us to take an approximate ones, perform the reduction of the numerator and denominator in half until the denominator becomes less than $k^2 = 256$:

   \[ r = \frac{A}{B} = \frac{1485}{793} \approx \frac{742}{396} \approx \frac{371}{198} \approx 1.873 \]

   By the asymptotic complexity theorem of an approximating $k$-ary GCD algorithm, the number of iterations at each stage is estimated to be $O(n/\log_2 k)$, where $n$ is the length of the origin numbers in bits. The complexity of the entire algorithm is estimated by the value

   \[ O(n^2/L + nL), \]

where $L$ is the binary length of $k$.

3. Experimental results and discussion

According to the results of our realization of the main cycle of an approximating k-ary GCD algorithm with parallelization of the calculation in C++ programming language in Microsoft Visual Studio 2012, we present our experimental data in the tables (1-3) and corresponding graphs (fig.1-5) with the choice of the optimal parameter $k$, a pre-computed the table of inverse elements and the search for $x$ and $y$ using Farey fractions ([9], chapter 2).

### Table 1. Evaluation of the average number of iterations and total time of AKA for 20 pairs of length 8500 decimal digits with different parameters $k$.

| No. | Parameter $k$ | Iterations | Time in mlseconds |
|-----|---------------|------------|-------------------|
| 1   | 64            | 431.9      | 906               |
| 2   | 256           | 274.9      | 768               |
| 3   | 1024          | 215.8      | 640               |
| 4   | 4096          | 208.8      | 594               |
| 5   | 16384         | 89.9       | 531               |
| 6   | 65536         | 94.5       | 516               |
We see that the optimal time is obtained at $k = 65536$ and the optimal iterations at $k = 16384$. This is portrayed in figure 1.

![AKA iteration and time](image1)

**Figure 1.** Evaluation of the average number of iterations and total time for AKA with different parameters $k$.

In figure 2 we see that we obtain the optimal iterations at $k = 16384$ and $k = 65536$.

![AKA Iteration](image2)

**Figure 2.** Evaluation of the average number of AKA iterations using different parameters $k$ for 20 pairs of length from 500 to 10500 decimal digits.

![AKA Time](image3)

**Figure 3.** Evaluation of the total time of AKA using different parameters $k$ for 20 pairs of length from 500 to 10500 decimal digits.
In figure 3 we see that we obtain the optimal time at \( k = 65536 \).

In tables 2 and 3 and in figures 4 and 5 we present a comparison of the AKA with other GCD algorithms and show that the AKA wins in average number of iterations and in total time for 20 pairs of numbers with length from 500 to 10500 decimal digits, at \( k = 65536 \).

**Table 2.** Evaluation of the average number of AKA iterations with Kary and CEA.

| digits | 500 | 1500 | 2500 | 3500 | 4500 | 5500 | 6500 | 7500 | 8500 | 9500 | 10500 |
|--------|-----|------|------|------|------|------|------|------|------|------|--------|
| CEA iteration | 977.9 | 2910.4 | 6796.8 | 8748.4 | 10701.4 | 12628.7 | 14559.5 | 16494 | 18441.5 | 20362 |
| Kary iteration | 666.6 | 2008.9 | 3351.1 | 4680.6 | 6028.6 | 7369.2 | 8705.8 | 10073.8 | 11366.1 | 12724.5 | 14087 |
| AKA iteration | 18.4 | 28.8 | 40.4 | 48.2 | 56.5 | 67 | 76.6 | 88.8 | 94.5 | 108 | 111.3 |

**Figure 4.** Evaluation of the average number of AKA iterations with Kary and CEA.

**Table 3.** Evaluation of the total time of AKA with Kary and CEA.

| digits | 500 | 1500 | 2500 | 3500 | 4500 | 5500 | 6500 | 7500 | 8500 | 9500 | 10500 |
|--------|-----|------|------|------|------|------|------|------|------|------|--------|
| CEA total time | 6 | 38 | 86 | 157 | 255 | 374 | 527 | 689 | 870 | 1078 | 1294 |
| Kary total time | 15 | 141 | 359 | 703 | 1141 | 1672 | 2312 | 3063 | 3922 | 4862 | 5922 |
| AKA total time | 0 | 32 | 62 | 94 | 156 | 234 | 313 | 406 | 516 | 609 | 750 |

**Figure 5.** Evaluation of the total time of AKA with Kary and CEA.
4. Conclusion
To reduce the number of iterations and speed up the operation of an approximating $k$-ary GCD algorithm for long numbers greater than 2000 decimal digits, we can choose the optimal parameter $k$ equal to the power of two. This will allow replacing the division operation by $k$ with a shift. We can also pre-calculate the table of inverse elements. It is also useful to search for suitable $x$ and $y$ using Farey fractions. For numbers of shorter length, the oldest version of the algorithm (classical Euclidean algorithm) is the most optimal.

References
[1] Ishmukhametov S T 2011 The factorization methods for natural numbers (Russia: Kazan) 190 p
[2] Sorenson J 2004 An analysis of the generalized binary GCD algorithm Lectures in Honour of Hugh Cowie Williams. – Banff, Alberta, Canada. – AMS Math. Review 41 254-8
[3] Sorenson J 1990 The k-ary GCD algorithm Computer Sciences Technical Report 20 p
[4] Sorenson J 1994 Two fast GCD Algorithms Journal of Algorithms 16(1) 110-44
[5] Amer Ismail 2018 On one acceleration of the $k$-ary GCD algorithm Scientific research of the SCO countries: synergy and integration pp 132-4
[6] Amer Ismail2018 Selecting the interval of the coefficient $y$ of the $k$-ary GCD algorithm for natural numbers Recent trend in Science and Technology management pp 12-5
[7] Amer Ismail 2020 On acceleration of the $k$-ary GCD algorithm IOP Conference Series: Materials Science and Engineering 734 012149
[8] Ishmukhametov S 2016 An approximating k-ary GCD algorithm Lobachevskii Journal of Mathematics 1 37(6) 723-9
[9] Hardy G H 1959 Wright E. M. An introduction to the theory of numbers (Oxford: Calrendon Press)