AN EFFICIENT BINARY ALGORITHM FOR SOLVING EQUATION $GCD * 2^{|J-K|} = X * A0 + Y * B0$

Viktor Matanski[1]

[1] *Faculty of Mathematics and Informatics, University of Plovdiv "Paisii Hilendarski", 24 Tzar Asen Str., 4000 Plovdiv, Bulgaria*
Email: matanski@uni-plovdiv.bg

Abstract. Using as a base the approach from [7-30] and the algorithm from [41] we present one fast binary algorithm for solving the equation $gcd * 2^{|j-k|} = x * a0 + y * b0$.

Key Words: *binary algorithm, reduced number of operations.*

2020 Mathematics Subject Classification: 11A05, 68W01

Main Results

Let a and b be integer numbers such as $a \geq 1$ and $b \geq 1$. Let us define $a0 = a$ and $b0 = b$. There are in existence integer numbers $j \geq 0$, $a2 \geq 1$, $k \geq 0$, $b2 \geq 1$ for which $a0 = a2 * 2^j$, $b0 = b2 * 2^k$, where a2 and b2 are odd numbers. Our algorithms below seek integer numbers x and y such that $gcd * 2^{|j-k|} = x * a0 + y * b0$, where gcd is the greatest common divisor of a and b. For classical algorithms, see [1-3] and [31-41]. New investigations, which concern efficient Euclidean algorithms, are presented in [7-30], [35-40].

Algorithm 1.

```
a0 = a; b0 = b; iter = 0;
x1 = 1; x2 = 0; y1 = 0; y2 = 1;
while ((a & 1) == 0)
{ iter++; a >>= 1; }
j = iter;
while ((b & 1) == 0)
{ iter++; b >>= 1; }
k = iter - j;
if (j < k) min = j; else min = k;
```

```
jk = j + k; a2 = a; b2 = b;
while (a != b)
if (a < b)
{
b -= a; y1 += x1; y2 += x2;
do
{ iter++; b >>= 1; x1 <<= 1; x2 <<= 1; }
while ((b & 1) == 0);
}
else
{
a -= b; x1 += y1; x2 += y2;
do
{ iter++; a >>= 1; y1 <<= 1; y2 <<= 1; }
while ((a & 1) == 0);
}
while (iter > jk)
{
if ((y1 & 1) == 1 ||(y2 & 1) == 1) { y1 += b2; y2 += a2; }
iter--; y1 >>= 1; y2 >>= 1;
}
eeacmi1 = gcd = a << min;
//as a result we receive:
//eeacmi1 = Greatest Common Divisor of a0 and b0
y1 = -y1; x = y1; y = y2;
if (j == min)
eeacmi2 = (x <<= (k - j)) * a0 + y * b0;
//as a result we receive: eeacmi2 = gcd * 2^{k-j} = x * a0 + y * b0
else
eeacmi2 = x * a0 + (y <<= (j - k)) * b0;
//as a result we receive: eeacmi2 = gcd * 2^{j-k} = x * a0 + y * b0
```

and its recursive version as

Algorithm 2.

```
static long Euclid(long a, long b,
ref int iter, long x1, long x2, ref long y1, ref long y2)
{
if ((a & 1) == 0)
{
iter++; y1 <<= 1; y2 <<= 1;
return Euclid(a >> 1, b, ref iter, x1, x2, ref y1, ref y2);
}
if ((b & 1) == 0)
{
iter++;
return Euclid(a, b >> 1, ref iter, x1 << 1, x2 << 1, ref y1, ref y2);
}
if (a == b) return a;
else
if (a > b)
return Euclid(a - b, b, ref iter, x1 + y1, x2 + y2, ref y1, ref y2);
else
{
y1 += x1; y2 += x2;
return Euclid(a, b - a, ref iter, x1, x2, ref y1, ref y2);
}
}
```

and its calling:

```
a0 = a; b0 = b; iter = 0;
x1 = 1; x2 = 0; y1 = 0; y2 = 1;
while ((a & 1) == 0)
{ iter++; a >>= 1; }
j = iter;
while ((b & 1) == 0)
{ iter++; b >>= 1; } k = iter - j;
if (j < k) min = j; else min = k;
jk = j + k; a2 = a; b2 = b;
gcd = Euclid(a2, b2, ref iter, x1, x2, ref y1, ref y2);
```

```
while (iter > jk)
{
if ((y1 & 1) == 1 ||(y2 & 1) == 1) { y1 += b2; y2 += a2; }
iter--; y1 >>= 1; y2 >>= 1;
}
eeacmi1 = gcd << min;
//as a result we receive:
//eeacmi1 = Greatest Common Divisor of a0 and b0
y1 = -y1; x = y1; y = y2;
if (j == min)
eeacmi2 = (x <<= (k - j)) * a0 + y * b0;
//as a result we receive: eeacmi2 = gcd * 2^{k-j} = x * a0 + y * b0
else
eeacmi2 = x * a0 + (y <<= (j - k)) * b0;
//as a result we receive: eeacmi2 = gcd * 2^{j-k} = x * a0 + y * b0
```

Numerical Example

For testing purposes we will use the following computer: processor - Intel(R) Core(TM) i7-6700HQ CPU 2.60GHz, 2592 Mhz, 4 Core(s), 8 Logical Processor(s), RAM 16 GB, Microsoft Windows 10 Enterprise x64, Microsoft Visual C# 2017 x64.

We will use the following example:

```
long a, b, gcd, eeacmi1, eeacmi2, d1 = 0, d2 = 0, x, y;
long b0, a0, x1, x2, y1, y2, a2, b2;
int min, iter, j, k, jk;
for (int i = 1; i < 100000001; i++)
{
b = i; a = 200000002 - i;
//here is placed the algorithm 1,
//calling of recursive algorithm 2
d1 += eeacmi1; d2 += eeacmi2;
}
Console.WriteLine (d1); Console.WriteLine (d2);
```

CPU time results are:

- CPU time of Algorithm 1 is: **89.287** seconds.
- CPU time of Algorithm 2 is: **200.377** seconds.

Acknowledgments

This paper is supported by the Project FP21-FMI-002 “Intelligent Innovative ICT in Research in Mathematics, Informatics and Pedagogy of Education” of the Scientific Fund of the University of Plovdiv “Paisii Hilendarski”, Bulgaria.

References

[1] S. Enkov, *Programming in Arduino Environment*, University Press “Paisii Hilendarski”, Plovdiv, 2017, (in Bulgarian).

[2] K. Garov, A. Rahnev, *Textbook-notes on programming in BASIC for facultative training in mathematics for 9.-10. grade of ESPU*, Sofia, 1986, (in Bulgarian).

[3] A. Golev, *Textbook on algorithms and programs in C#*, University Press “Paisii Hilendarski”, Plovdiv, 2012, (in Bulgarian).

[4] T. Terzieva, *Introduction to web programming*, University Press “Paisii Hilendarski”, Plovdiv, 2021, ISBN: 978-619-202-623-3, (in Bulgarian).

[5] T. Terzieva, *Development of algorithmic thinking in the Informatics Education*, University Press “Paisii Hilendarski”, Plovdiv, 2021, ISBN: 978-619-202-622-6, (in Bulgarian).

[6] T. Terzieva, *Educational tools for teaching in digital environment*, University Press “Paisii Hilendarski”, Plovdiv, 2021, (in Bulgarian).

[7] A. Iliev, N. Kyurkchiev, A Note on Knuth’s Implementation of Euclid’s Greatest Common Divisor Algorithm, *International Journal of Pure and Applied Mathematics*, 117, 2017, 603–608.

[8] A. Iliev, N. Kyurkchiev, A. Golev, A Note on Knuth’s Implementation of Extended Euclidean Greatest Common Divisor Algorithm, *International Journal of Pure and Applied Mathematics*, 118, 2018, 31–37.

[9] A. Iliev, N. Kyurkchiev, A. Rahnev, A Note on Adaptation of the Knuth’s Extended Euclidean Algorithm for Computing Multiplicative Inverse, *International Journal of Pure and Applied Mathematics*, 118, 2018, 281–290.

[10] A. Iliev, N. Kyurkchiev, A Note on Euclidean and Extended Euclidean Algorithms for Greatest Common Divisor for Polynomials, *International Journal of Pure and Applied Mathematics*, 118, 2018, 713–721.

[11] A. Iliev, N. Kyurkchiev, *New Trends in Practical Algorithms: Some Computational and Approximation Aspects*, LAP LAMBERT Academic Publishing, Beau Bassin, 2018.

[12] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement Euclidean Algorithm for Greatest Common Divisor. I, *Neural, Parallel, and Scientific Computations*, 26, No. 3, 2018, 355–362.

[13] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Least Absolute Remainder Algorithm for Greatest Common Divisor. III, *Neural, Parallel, and Scientific Computations*, 27, No. 1, 2019, 1–9.

[14] A. Iliev, N. Kyurkchiev, A. Rahnev, *Nontrivial Practical Algorithms: Part 2*, LAP LAMBERT Academic Publishing, Beau Bassin, 2019.

[15] A. Iliev, N. Valchanov, T. Terzieva, Generalization and Optimization of Some Algorithms, *Collection of scientific works of National Conference "Education in Information Society"*, Plovdiv, ADIS, 12-13 May 2009, (2009), 52–58, (in Bulgarian).

[16] H. Gyulyustan, A Note on Euclidean Sequencing Algorithm, *Proceedings of the Scientific Conference "Innovative ICT for Digital Research Area in Mathematics, Informatics and Pedagogy of Education"*, Pamporovo, 7-8 November 2019, Plovdiv University Press, 2020, 57–64.

[17] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Stein's Binary Algorithm for Finding Greatest Common Divisor, *Neural, Parallel, and Scientific Computations*, 28, No. 1, 2020, 75–80.

[18] A. Iliev, N. Kyurkchiev, A. Rahnev, New Algorithms for Finding Modular Multiplicative Inverse, *Neural, Parallel, and Scientific Computations*, 28, No. 1, 2020, 81–88.

[19] A. Iliev, N. Kyurkchiev, A. Rahnev, New Extended Algorithm for Finding Greatest Common Divisor, *Neural, Parallel, and Scientific Computations*, 28, No. 1, 2020, 89–95.

[20] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Modular Multiplicative Inverse Binary Algorithm, *International Electronic Journal of Pure and Applied Mathematics*, 14, No. 1, 2020, 37–44.

[21] A. Iliev, N. Kyurkchiev, A. Rahnev, Recursive Extended Stein's Binary Algorithm, *International Electronic Journal of Pure and Applied Mathematics*, 14, No. 1, 2021, 31–36.

[22] A. Iliev, N. Kyurkchiev, A. Rahnev, A new improvement of Jacobi symbol algorithm, *International Electronic Journal of Pure and Applied Mathematics*, 15, No. 1, 2021, 13–22.

[23] A. Iliev, N. Kyurkchiev, A. Rahnev, A new improvement of Jacobi symbol binary algorithm, *International Electronic Journal of Pure and Applied Mathematics*, 15, No. 1, 2021, 1–11.

[24] A. Iliev, N. Kyurkchiev, A. Rahnev, Efficient Binary Algorithm for Kronecker Symbol, *Communications in Applied Analysis*, 25, No. 1, 2021, 11–21.

[25] A. Iliev, N. Kyurkchiev, A. Rahnev, Efficient Algorithm for Kronecker Symbol, *International Electronic Journal of Pure and Applied Mathematics*, 15, No. 1, 2021, 23–30.

[26] A. Iliev, N. Kyurkchiev, A. Rahnev, T. Terzieva, A Refinement of the Extended Euclidean Algorithm using SGN Function, *Communications in Applied Analysis*, 25, No. 1, 2021, 39–51.

[27] A. Iliev, N. Kyurkchiev, A. Rahnev, A Refinement of the Knuth's Extended Euclidean Algorithm for Computing Modular Multiplicative Inverse, *Communications in Applied Analysis*, 25, No. 1, 2021, 23–37.

[28] A. Iliev, N. Kyurkchiev, A. Rahnev, A Refinement of the Extended Euclidean Algorithm, *International Electronic Journal of Pure and Applied Mathematics*, 15, No. 1, 2021, 33–44.

[29] A. Iliev, N. Kyurkchiev, A. Rahnev, T. Terzieva, A Refinement of the Böh's Algorithm for Computing Modular Multiplicative Inverse, *International Electronic Journal of Pure and Applied Mathematics*, 15, No. 1, 2021, 45–53.

[30] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Extended Stein's Binary Algorithm, *Proceedings of the Anniversary International Scientific Conference "Synergetics and Reflection in Mathematics Education"*, Pamporovo, 16-18 October 2020, Plovdiv University Press, 2020, 259–264.

[31] A. Rahnev, K. Garov, O. Gavrailov, *Textbook for extracurricular work using BASIC*, MNP Press, Sofia, 1985, (in Bulgarian).

[32] A. Rahnev, K. Garov, O. Gavrailov, *BASIC in examples and tasks*, Government Press "Narodna prosveta", Sofia, 1990, (in Bulgarian).

[33] N. Kasakliev, *C# Programming Guide*, University Press "Paisii Hilendarski", Plovdiv, 2016, (in Bulgarian).

[34] A. Rahnev, N. Pavlov, N. Valchanov, T. Terzieva, *Object Oriented Programming*, Lightning Source UK Ltd., London, 2014.

[35] D. Rachmawati, M. Budiman, On Using the First Variant of Dependent RSA Encryption Scheme to Secure Text: A Tutorial, *J. Phys.: Conf. Ser.*, 2020, 1542 012024.

[36] J. Erho, J. Consul, B. Japheth, Juggling Versus Three-Way-Reversal Sequence Rotation Performance Across Four Data Types, *International Journal of Scientific Research in Computer Science and Engineering*, 7, No. 6, 2019, 10–18.

[37] J. Butar-butar, F. Sinuhaji, Faktorisasi Polinomial Square-Free dan bukan Square-Free atas Lapangan Hingga Zp, *Jurnal Teori dan Aplikasi Matematika*, 3, No. 2, 2019, 132–142.

[38] L. Akcay, B. Ors, Comparison of RISC-V and transport triggered architectures for a post-quantum cryptography application, *Turk J Elec Eng & Comp Sci*, 29, 2021, 321–333.

[39] C. Falcon Rodriguez, M. Cruz, C. Falcon, Full Euclidean Algorithm by Means of a Steady Walk, *Applied Mathematics*, 12, 2021, 269–279.

[40] Y. Fan, G. Chen, M. Cui, Formalization of Finite Field GF(2^n) based on COQ, *Computer Science*, 47, No. 12, 2020, 311–318.

[41] F. Böh, Verfahren zur Berechnung der modularen Inversen zweier Zahlen, *European Patent Office*, EP 1 271 304 A2, 2003, 11 pp.