International Journal of Differential Equations and Applications
Volume **20**, No. 2 (2021), pages: 179-186
ISSN (Print): 1311-2872; ISSN (Online): 1314-6084;
url: https://www.ijdea.eu
ijdea.eu

EFFICIENT BINARY EXTENDED ALGORITHM USING SGN FUNCTION

Anton Iliev[1], Nikolay Kyurkchiev[2], Asen Rahnev[3], Todorka Terzieva[4]
[1,2,3,4]Faculty of Mathematics and Informatics
Paisii Hilendarski University of Plovdiv
24 Tzar Asen Str., 4000 Plovdiv, BULGARIA
[1,2]Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8, 1113 Sofia, BULGARIA

ABSTRACT: We present new binary extended algorithms that work for every integer numbers a and b for which $a \neq 0$ and $b \neq 0$. The approach given here generalizes and optimizes the algorithm given in the monograph of A. Menezes, P. Oorschot and S. Vanstone [39] as well our results from [28] and [20]. These computation ways demonstrate high computational effectiveness especially for long numbers.

AMS Subject Classification: 11A05, 68W01
Key Words: binary extended algorithm, sgn function, reduced number of operations

Received: August 20, 2021 **Revised:** November 4, 2021
Published: November 15, 2021 **doi:** 10.12732/ijdea.v20i2.4
Academic Publications, Ltd. https://acadpubl.eu

1. MAIN RESULTS

For two integer numbers a and b such that $a \neq 0$, $b \neq 0$ we construct binary iteration process which find integer numbers x and y so that $x * a + y * b =$ *greatest common divisor (gcd)*. The new applications of the class of Euclidean algorithms can be found in [33]–[38].

Contradictorily to so-called traditional approach [1]–[6], [30]–[32], [39], which is widely spread in numerous papers and books again we present the efficient computational way [7]–[29] specifically for the task presented in this article. Our iteration processes [7]–[29] give better computational speed performance because in them some unnecessary operations are economized uncovering the natural algorithmic way for realization of such algorithms.

For illustrating our results we will use Microsoft Windows 10 Enterprise x64 and Microsoft Visual C# 2017 x64. For that purpose we propose:

Algorithm 1.

```
g = 0; x2 = 0; y1 = 0;
if (a > 0) x1 = 1; else if (a < 0) x1 = -1;
if (b > 0) y2 = 1; else if (b < 0) y2 = -1;
sng = x1 * y2;
b = Math.Abs(b); a = Math.Abs(a);

if ((a & 1) == 0 && (b & 1) == 0)
    do { a >>= 1; b >>= 1; g++; }
    while ((a & 1) == 0 && (b & 1) == 0);

u = a; v = b;
while ((u & 1) == 0)
{
    u >>= 1;
    if ((x1 & 1) == 0 && (x2 & 1) == 0) { x1 >>= 1; x2 >>= 1; }
    else if (sng > 0) { x1 = (x1 + b) >> 1; x2 = (x2 - a) >> 1; }
    else { x1 = (x1 - b) >> 1; x2 = (x2 - a) >> 1; }
}
while ((v & 1) == 0)
{
    v >>= 1;
    if ((y1 & 1) == 0 && (y2 & 1) == 0) { y1 >>= 1; y2 >>= 1; }
    else if (sng > 0) { y1 = (y1 + b) >> 1; y2 = (y2 - a) >> 1; }
    else { y1 = (y1 - b) >> 1; y2 = (y2 - a) >> 1; }
}
while (u != v)
    if (u > v)
    {
        u -= v; x1 -= y1; x2 -= y2;
        do
        {
            u >>= 1;
            if ((x1 & 1) == 0 && (x2 & 1) == 0) { x1 >>= 1; x2 >>= 1; }
            else if (sng > 0) { x1 = (x1 + b) >> 1; x2 = (x2 - a) >> 1;
```

```
    }
            else { x1 = (x1 - b) >> 1; x2 = (x2 - a) >> 1; }
        } while ((u & 1) == 0);
    }
    else
    {
        v -= u; y1 -= x1; y2 -= x2;
        do
        {
            v >>= 1;
            if ((y1 & 1) == 0 && (y2 & 1) == 0) { y1 >>= 1; y2 >>= 1; }
            else if (sng > 0) { y1 = (y1 + b) >> 1; y2 = (y2 - a) >> 1;
    }
            else { y1 = (y1 - b) >> 1; y2 = (y2 - a) >> 1; }
        } while ((v & 1) == 0);
    }
x = y1; y = y2; gcd = v << g;
```

and its recursive variant:

Algorithm 2.

```
static long Euclid(long a0, long b0, long a, long b,
long sng, long x1, long x2, ref long y1, ref long y2)
    {
         if ((a & 1) == 0)
    {
 if ((b & 1) == 0)
    return Euclid(a0 >> 1, b0 >> 1, a >> 1, b >> 1,
    sng, x1, x2, ref y1, ref y2) << 1;
 else
     {
    if ((x1 & 1) == 0 && (x2 & 1) == 0)
    { x1 >>= 1; x2 >>= 1; }
    else if (sng > 0) { x1 = (x1 + b0) >> 1; x2 = (x2 - a0) >> 1; }
    else { x1 = (x1 - b0) >> 1; x2 = (x2 - a0) >> 1; }
    return Euclid(a0, b0, a >> 1, b, sng, x1, x2, ref y1, ref y2);
    }
    }
    else if ((b & 1) == 0)
    {
 if ((y1 & 1) == 0 && (y2 & 1) == 0)
 { y1 >>= 1; y2 >>= 1; }
 else if (sng > 0) { y1 = (y1 + b0) >> 1; y2 = (y2 - a0) >> 1; }
 else { y1 = (y1 - b0) >> 1; y2 = (y2 - a0) >> 1; }
 return Euclid(a0, b0, a, b >> 1, sng, x1, x2, ref y1, ref y2);
     }
```

```
      else
      if (a == b) return a;
        else
        if (a > b)
 return Euclid(a0, b0, a - b, b, sng, x1 - y1, x2 - y2, ref y1, ref y2);

        else
        {
 y1 -= x1; y2 -= x2;
 return Euclid(a0, b0, a, b - a, sng, x1, x2, ref y1, ref y2);
        }
        }
```

and its calling:

```
x2 = 0; y1 = 0;
if (a > 0) x1 = 1; else if (a < 0) x1 = -1;
if (b > 0) y2 = 1; else if (b < 0) y2 = -1;
sng = x1 * y2;
b = Math.Abs(b); a = Math.Abs(a);
a0 = a; b0 = b;
gcd = Euclid(a0, b0, a, b, sng, x1, x2, ref y1, ref y2);
x = y1; y = y2;
```

2. NUMERICAL EXPERIMENT

For testing our approach we will use the following computer configuration: processor – Intel(R) Core(TM) i7-6700HQ CPU 2.60 GHz, 2592 Mhz, 4 Core(s), 8 Logical Processor(s), RAM 16 GB.

```
long a, b, gcd, d = 0, x, y;
long a0, b0, x1 = 0, x2, y1, y2 = 0, u, v, sng;
int g;
for (int i = 1; i < 100000001; i++)
{
a = i; b = 200000002 - i;
//here can be placed the algorithm 1
//and the calling of recursive algorithm 2
d += gcd;
}
Console.WriteLine (d);
```

CPU time of Algorithm 1 is 72.340 seconds.

CPU time of Algorithm 2 is 201.964 seconds.

ACKNOWLEDGMENTS

This work has been accomplished with the financial support by the Grant No BG05M2O P001-1.001-0003, financed by the Science and Education for Smart Growth Operational Program (2014-2020) and co-financed by the European Union through the European structural and Investment funds.

REFERENCES

[1] S. Enkov, *Programming in Arduino Environment*, Paisii Hilendarski University Press, 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#*, Paisii Hilendarski University Press, Plovdiv, (2012), (in Bulgarian).

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

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

[6] T. Terzieva, *Educational tools for teaching in digital environment*, Paisii Hilendarski University Press, 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*, (2017), **117**, 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*, (2018), **118**, 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*, (2018), **118**, 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*, (2018), **118**, 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*, (2018), **26**, No. 3, 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, (2019), **27**, No. 1, 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] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Stein's Binary Algorithm for Finding Greatest Common Divisor, *Neural, Parallel, and Scientific Computations*, (2020), **28**, No. 1, 75–80.

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

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

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

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

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

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

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

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

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

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

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

[28] 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.

[29] V. Matanski, An Efficient Binary Algorithm for Solving Equation $gcd * 2^{|j-k|} = x * a0 + y * b0$, *Proceedings of the Anniversary International Scientific Conference "Computer Technologies and Applications"*, Pamporovo, 15–17 September 2021, Plovdiv University Press, (2021), 79–86.

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

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

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

[33] 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.

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

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

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

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

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

[39] A. Menezes, P. Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, 5th ed., CRC Press LLC, New York, (2001).