Neural, Parallel, and Scientific Computations, **28**, No. 1 (2020), 81-89 ISSN: 1061-5369

NEW ALGORITHMS FOR FINDING MODULAR MULTIPLICATIVE INVERSE

Anton Iliev[1], Nikolay Kyurkchiev[2], Asen Rahnev[3]

[1,2,3]Faculty of Mathematics and Informatics
University of Plovdiv Paisii Hilendarski
24, Tzar Asen Str., 4000 Plovdiv, BULGARIA

ABSTRACT: In this paper we present several algorithms for finding modular multiplicative inverse. These algorithms are based on our previous research. They are appropriate for regular as well as for long numbers. In many papers [7]–[33] we present new iterative and recursive schemes for various issues that concern the classical task of finding greatest common divisor. The algorithms in this paper are based on combination of "remainder" and "difference" operations. The reason of this interest is because for long numbers the CPU Time of "remainder" operation is too high. The presented algorithms are given in optimal or near to optimal computational ways, they are highly symmetrized, work in both iterative and recursive cases and continue the results in [7]–[33]. The invented algorithms do not depend on both the hardware computer system and the software environment in which they can be implemented.

AMS Subject Classification: 11A05, 68W01
Key Words: modular multiplicative inverse, recursive algorithm, iterative algorithm, reduced number of operations

Received: May 20, 2020; **Accepted:** October 5, 2020;
Published: October 17, 2020. **doi:** 10.12732/npsc.v28i1.9
Dynamic Publishers, Inc., Acad. Publishers, Ltd. https://acadsol.eu/npsc

1. INTRODUCTION

Our new algorithms work correctly for every $a > 0$ and $b > 0$. They both iterative and recursive compute the multiplicative inverse of a *modulo* b; $a^{-1} \pmod{b}$, and returns either the inverse as a positive integer less than b, or 0 if no inverse exists. The multiplicative inverse of a *modulo* b exists if and only if a and b are coprime (i.e., if their greatest common divisor (gcd) is equal to 1). We give the computational way

which is different in comparison to computational manner which is given in [1]–[6] and [32]–[40].

For testing purposes for new algorithm 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.

Main Results

We propose the following iterative algorithm for finding modular multiplicative inverse –

Algorithm 1.

```
x1 = 1; x2 = 0; b0 = b;
while (b > 0)
{ q = a / b; r = a % b; a = b; b = r;
t = x2; x2 = x1 - q * x2; x1 = t; }
if (a == 1)
{ if (x1 < 0) x1 += b0;
eeacmi = x1; }
else eeacmi = 0;
```

and its recursive variant –

Algorithm 2.

```
static long Euclid(long a, long b, ref long x, ref long y)
{ if (b < 1) { x = 1; y = 0; return a; }
long q = a / b; long r = a % b;
long d = Euclid(b, r, ref y, ref x);
y -= q * x;
return d; }
```

It can be called by:

```
b0 = b;
if (a > b) gcd = Euclid(a, b, ref x, ref y);
else gcd = Euclid(b, a, ref y, ref x);
if (gcd > 1) eeacmi = 0;
else { while (x < 0) x += b; while (x >= b) x -= b; eeacmi = x; }
```

We introduce the following optimized iterative –

Algorithm 3.

```
b0 = b;
if (a > b) { x1 = 1; x2 = 0;
do { q = a / b; a %= b; t = x2; x2 = x1 - q * x2; x1 = t;
if (a < 1) { gcd = b; break; }
q = b / a; b %= a; t = x2; x2 = x1 - q * x2; x1 = t;
if (b < 1) { gcd = a; break; }
} while (true); }
else { x1 = 0; x2 = 1;
do { q = b / a; b %= a; t = x2; x2 = x1 - q * x2; x1 = t;
if (b < 1) { gcd = a; break; }
q = a / b; a %= b; t = x2; x2 = x1 - q * x2; x1 = t;
if (a < 1) { gcd = b; break; }
} while (true); }
if (gcd == 1) { if (x1 < 0) x1 += b0; eeacmi = x1; }
else eeacmi = 0;
```

and the following optimized recursive –

Algorithm 4.

```
static long Euclid(long a, long b, ref long x, ref long y)
{ long r = a % b; long q1 = a / b;
if (r < 1) { x = 1; y = 0; return b; }
long u = b % r; long q2 = b / r;
if (u < 1) { x = -q1; y = 1; return r; }
long d = Euclid(r, u, ref x, ref y);
y -= q2 * x; x -= q1 * y;
return d; }
```

It can be called by:

```
b0 = b;
if (a > b) gcd = Euclid(a, b, ref y, ref x);
else gcd = Euclid(b, a, ref x, ref y);
if (gcd == 1) { if (x < 0) x += b0;
```

```
eeacmi = x; }
else eeacmi = 0;
```

We introduce the following iterative

Algorithm 5.

```
b0 = b; x1 = 1; x2 = 0;
do
if (a > b) { q = a / b; a %= b; t = x2; x2 = x1 - q * x2; x1 = t;
if (a < 1) { gcd = b; x = x1; break; }
else { b -= a; t = x2; x2 = x1 - x2; x1 = t;
if (a == b) { gcd = a; x = x1; break; } } }
else { q = b / a; b %= a; t = x1; x1 = x2 - q * x1; x2 = t;
if (b < 1) { gcd = a; x = x2; break; }
else { a -= b; t = x1; x1 = x2 - x1; x2 = t;
if (a == b) { gcd = b; x = x2; break; } } }
while (true);
if (gcd == 1) { if (x < 0) x += b0;
eeacmi = x; }
else eeacmi = 0;
```

and its recursive analog as

Algorithm 6.

```
static long Euclid(long a, long b, ref long x, ref long y)
{ long r = a % b; long q1 = a / b;
if (r < 1) { x = 1; y = 0; return b; }
long u = b - r;
if (u == r) { x = -q1; y = 1; return r; }
long d;
if (r > u) d = Euclid(r, u, ref x, ref y);
else d = Euclid(u, r, ref y, ref x);
y -= x; x -= q1 * y;
return d; }
```

It can be called by:

```
b0 = b;
if (a > b) gcd = Euclid(a, b, ref y, ref x);
else gcd = Euclid(b, a, ref x, ref y);
```

```
if (gcd == 1) { if (x < 0) x += b0;
eeacmi = x; }
else eeacmi = 0;
```

Numerical Example

We will test the proposed algorithms for the following example:

```
long a, b, x = 0, y = 0, d = 0, b0, q, r, t;
long x1, x2, gcd, eeacmi;

for (int i = 1; i < 100000001; i++) { a = i; b = 200000002 - i;
//Here can be placed the source codes of Algorithms 1, 3 and 5
//and the calling of their recursive variants – Algorithm 2, 4 and 6.

d += eeacmi; }
Console.WriteLine(d);
```

CPU time of Algorithm 1 is: **34.639 seconds.**
CPU time of Algorithm 2 is: **61.432 seconds.**
CPU time of Algorithm 3 is: **30.819 seconds.**
CPU time of Algorithm 4 is: **45.935 seconds.**
CPU time of Algorithm 5 is: **32.597 seconds.**
CPU time of Algorithm 6 is: **52.820 seconds.**

Conclusion

We present several new algorithms in both styles – modular multiplicative inverse recursive and modular multiplicative inverse iterative respectively. The question of optimality for computational way and number of used operations will be considered by us on a next stage.

Acknowledgments

This paper is supported by the Project FP19-FMI-002 "Innovative ICT for Digital Research Area in Mathematics, Informatics and Pedagogy of Education" of the Scientific Fund of the University of Plovdiv Paisii Hilendarski, Bulgaria.

REFERENCES

[1] A. Akritas, A new method for computing polynomial greatest common divisors and polynomial remainder sequences, *Numerische Mathematik*, 52 (1988), 119–127.

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

[3] F. Chang, Factoring a Polynomial with Multiple-Roots, *World Academy of Science, Engineering and Technology*, 47 (2008), 492–495.

[4] Th. Cormen, Ch. Leiserson, R. Rivest, Cl. Stein, *Introduction to Algorithms*, 3rd ed., The MIT Press, Cambridge (2009).

[5] 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)

[6] A. Golev, *Textbook on algorithms and programs in C#*, University Press "Paisii Hilendarski", Plovdiv (2012). (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, A Note on Least Absolute Remainder Euclidean Algorithm for Greatest Common Divisor, *International Journal of Scientific Engineering and Applied Science*, 4 No. 3 (2018), 31–34.

[12] A. Iliev, N. Kyurkchiev, A Note on Knuth's Algorithm for Computing Extended Greatest Common Divisor using SGN Function, *International Journal of Scientific Engineering and Applied Science*, 4 No. 3 (2018), 26–29.

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

[14] A. Iliev, N. Kyurkchiev, 80th Anniversary of the birth of Prof. Donald Knuth, *Biomath Communications*, 5 (2018), 7 pp.

[15] A. Iliev, N. Kyurkchiev, New Realization of the Euclidean Algorithm, *Collection of scientific works of Eleventh National Conference with International Participation Education and Research in the Information Society*, Plovdiv, ADIS, June 1–2, (2018), 180–185. (in Bulgarian)

[16] A. Iliev, N. Kyurkchiev, New Organizing of the Euclid's Algorithm and one of its Applications to the Continued Fractions, *Collection of scientific works from conference "Mathematics. Informatics. Information Technologies. Application in Education"*, Pamporovo, Bulgaria, 10–12 October 2018, (2019), 199–207.

[17] A. Iliev, N. Kyurkchiev, The faster Euclidean algorithm, *Collection of scientific works from conference*, Pamporovo, Bulgaria, 28–30 November 2018, (2019), 15–20.

[18] A. Iliev, N. Kyurkchiev, The faster extended Euclidean algorithm, *Collection of scientific works from conference*, Pamporovo, Bulgaria, 28–30 November 2018, (2019), 21–26.

[19] P. Kyurkchiev, V. Matanski, The faster Euclidean algorithm for computing polynomial multiplicative inverse, *Collection of scientific works from conference*, Pamporovo, Bulgaria, 28–30 November 2018, (2019), 43–48.

[20] V. Matanski, P. Kyurkchiev, The faster Lehmer's greatest common divisor algorithm, *Collection of scientific works from conference, Pamporovo*, Bulgaria, 28–30 November 2018, (2019), 37–42.

[21] 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.

[22] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Harris–Stein Modification of Euclidean Algorithm for Greatest Common Divisor. II, *International Journal of Pure and Applied Mathematics*, 120 No. 3 (2018), 379–388.

[23] 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.

[24] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Tembhurne–Sathe Modification of Euclidean Algorithm for Greatest Common Divisor. IV, *Dynamic Systems and Applications*, 28 No. 1 (2019), 143–152.

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

[26] 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)

[27] H. Gyulyustan, A Note on Euclidean Sequencing Algorithm, *Proceedings of the Scientific Conference "Innovative ICT for Digital Research Area in Mathemat-*

ics, Informatics and Pedagogy of Education", Pamporovo, 7–8 November 2019, Plovdiv University Press, 2020, 57–64.

[28] A. Iliev, N. Kyurkchiev, A. Rahnev, New Algorithm for Finding Greatest Common Divisor, *preprint*.

[29] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Stein's Binary Algorithm for Finding Greatest Common Divisor, *preprint*.

[30] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Extended Stein's Binary Algorithm, *preprint*.

[31] A. Iliev, N. Kyurkchiev, A. Rahnev, Recursive Extended Stein's Binary Algorithm, *preprint*.

[32] A. Iliev, N. Kyurkchiev, A. Rahnev, New Extended Algorithm for Finding Greatest Common Divisor, *preprint*.

[33] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Modular Multiplicative Inverse Binary Algorithm, *preprint*.

[34] D. Knuth, *The Art of Computer Programming*, *Vol. 2, Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Boston (1998).

[35] Hr. Krushkov, A. Iliev, *Practical programming guide in Pascal, Parts I and II*, Koala press, Plovdiv (2002). (in Bulgarian)

[36] P. Nakov, P. Dobrikov, *Programming=++Algorithms*, 5th ed., Sofia (2015). (in Bulgarian)

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

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

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

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

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