Dynamic Systems and Applications, **28**, No. 1 (2019), 143-152 ISSN: 1056-2176

A NEW IMPROVEMENT OF TEMBHURNE-SATHE MODIFICATION OF EUCLIDEAN ALGORITHM FOR GREATEST COMMON DIVISOR. IV

ANTON ILIEV[1], NIKOLAY KYURKCHIEV[2], AND 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 note we gave new interpretation of Tembhurne-Sathe modification of Euclidean algorithm for calculation of greatest common divisor (GCD). Our results are different optimized ways of approaches presented in [1]–[26], [44]–[68]. Our approach is about 9% and 69% faster than Tembhurne-Sathe algorithm in iterative and recursive implementations respectively. For computer implementation Visual C# 2017 programming environment is used.

AMS Subject Classification: 11A05, 68W01
Key Words: greatest common divisor, Euclid's algorithm, Tembhurne-Sathe' algorithm, improvement algorithm, 256-bit integers, 512-bit integers, reduced number of iterations

Received: October 17, 2018 ; **Revised:** January 9, 2019 ;
Published (online): January 19, 2019 **doi:** 10.12732/dsa.v28i1.8
Dynamic Publishers, Inc., Acad. Publishers, Ltd. https://acadsol.eu/dsa

1. INTRODUCTION

Our modifications are based on ideas in [27]–[43]. Here we will describe possible way of optimization of Tembhurne-Sathe' algorithm for GCD. In [68] the authors note that their ModEB algorithm is about 1.54 and 1.68 times faster than the Extended Euclidean and Binary GCD algorithm with 256-bit integers as well as it is about 1.54 and 1.64 times faster than the same algorithms with 512-bit integers. Out motivation here is to give more optimal organization of Tembhurne-Sathe' algorithm.

2. MAIN RESULTS

For testing 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.

Let $a>0$ and $b>0$ be natural numbers. Tembhurne-Sathe' algorithm [68] is well-known:

Algorithm 1.

```
static long Euclid(long a, long b)
{ long g = 1;
if (a == b) return a;
if ((a & 1) == 0 && (b & 1) == 0)
do { a >>= 1; b >>= 1; g <<= 1;
if (b == 1) return g; } while ((a & 1) == 0 && (b & 1) == 0);
while (b > 1) { a %= b; b = Math.Abs(b - a);
if (a < b) { long tmp = a; a = b; b = tmp; }
if ((a & 1) == 0) do { a >>= 1; if (a == 1) return g; }
while ((a & 1) == 0);
if ((b & 1) == 0) do { if (b == 0) break; b >>= 1; }
while ((b & 1) == 0); }
if (b == 1) return g; else return g * a; }
```

Its recursive implementation is:

Algorithm 2.

```
static long Euclid(long a, long b)
{ if (a == b) return a;
if ((a & 1) == 0 && (b & 1) == 0)
return Euclid(a >> 1, b >> 1) << 1;
a %= b; b = Math.Abs(b - a);
if (a > b) { if ((a & 1) == 0) { a >>= 1; if (a == 1) return 1; }
if ((b & 1) == 0) { if (b == 0) return a; b >>= 1; } }
else { if ((b & 1) == 0) { b >>= 1; if (b == 1) return 1; }
if ((a & 1) == 0) { if (a == 0) return b; a >>= 1; } }
return Euclid(a, b); }
```

We suggest the following optimized realization of Algorithm 1

Algorithm 3.

```
static long Euclid(long a, long b)
{ int k = 0, g = 1;
```

```
if ((a & 1) == 0 && (b & 1) == 0) do
{ a >>= 1; b >>= 1; k++; }
while ((a & 1) == 0 && (b & 1) == 0);
g <<= k;
do { if (a > b) { a %= b; b = b - a;
if ((b & 1) == 0) do { b >>= 1; if (b == 1) return g; }
while ((b & 1) == 0);
if ((a & 1) == 0) do { if (a == 0) return b << k; a >>= 1; }
while ((a & 1) == 0);
b %= a; a = a - b;
if ((a & 1) == 0) do { a >>= 1; if (a == 1) return g; }
while ((a & 1) == 0);
if ((b & 1) == 0) do { if (b == 0) return a << k; b >>= 1; }
while ((b & 1) == 0); }
else { b %= a; a = a - b;
if ((a & 1) == 0) do { a >>= 1; if (a == 1) return g; }
while ((a & 1) == 0);
if ((b & 1) == 0) do { if (b == 0) return a << k; b >>= 1; }
while ((b & 1) == 0);
a %= b; b = b - a;
if ((b & 1) == 0) do { b >>= 1; if (b == 1) return g; }
while ((b & 1) == 0);
if ((a & 1) == 0) do { if (a == 0) return b << k; a >>= 1; }
while ((a & 1) == 0); }
} while (true); }
```

and its recursive presentation

Algorithm 4.

```
static long Euclid(long a, long b)
{ if ((a & 1) == 0 && (b & 1) == 0)
return Euclid(a >> 1, b >> 1) << 1;
if (a > b) { a %= b; b = b - a;
if ((b & 1) == 0) { b >>= 1; if (b == 1) return 1; }
if ((a & 1) == 0) { if (a == 0) return b; a >>= 1; }
b %= a; a = a - b;
if ((a & 1) == 0) { a >>= 1; if (a == 1) return 1; }
if ((b & 1) == 0) { if (b == 0) return a; b >>= 1; } }
else { b %= a; a = a - b;
if ((a & 1) == 0) { a >>= 1; if (a == 1) return 1; }
if ((b & 1) == 0) { if (b == 0) return a; b >>= 1; }
```

```
a %= b; b = b - a;
if ((b & 1) == 0) { b >>= 1; if (b == 1) return 1; }
if ((a & 1) == 0) { if (a == 0) return b; a >>= 1; } }
return Euclid(a, b); }
```

3. NUMERICAL EXPERIMENT

Part 1. This part is implemented for all Algorithms 1–4.

```
long a, b, gcd, d = 0;
for (int i = 1; i < 100000001; i++) { b = i; a = 200000002 - i;
//here is the calling of every of Algorithms 1–4
d += gcd; }
Console.WriteLine(d);
```

Part 2. This part is implemented for Algorithms 1 and 2 only! We will use the task from Part 1. where we swapped the values of 'a' and 'b'.

Part 3. Average time of performance for Algorithms 1 and 2 only!

EN = (Part 1.Algorithm$N$ + Part 2.Algorithm$N$) / 2,
where $N = 1$ to 2 denotes using of Algorithms 1 and 2.

Recursive and iterative implementation of algorithms 1 and 2 can be called by:

```
if (a > b) gcd = Euclid(a, b); else gcd = Euclid(b, a);
```

Recursive and iterative implementations of algorithms 3 and 4 can be called by:

```
gcd = Euclid(a, b);
```

The reader can see advantages of our new realizations Algorithm 4 and Algorithm 3 in both recursive (see Fig. 1) and iterative (see Fig. 2) implementations.

ACKNOWLEDGMENTS

This work has been supported by D01-205/23.11.2018 National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science, Bulgaria.

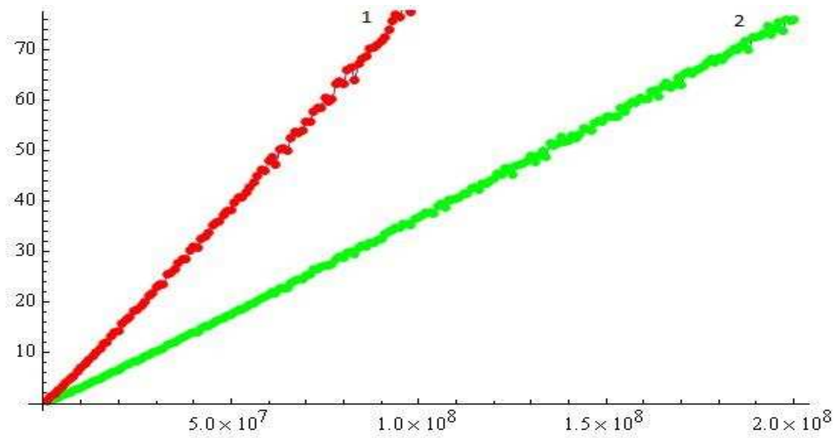

Figure 1: Algorithm 2 - Tembhurne-Sathe recursive (1 – red color), Algorithm 4 - Iliev–Kyurkchiev–Rahnev recursive (2 – green line)

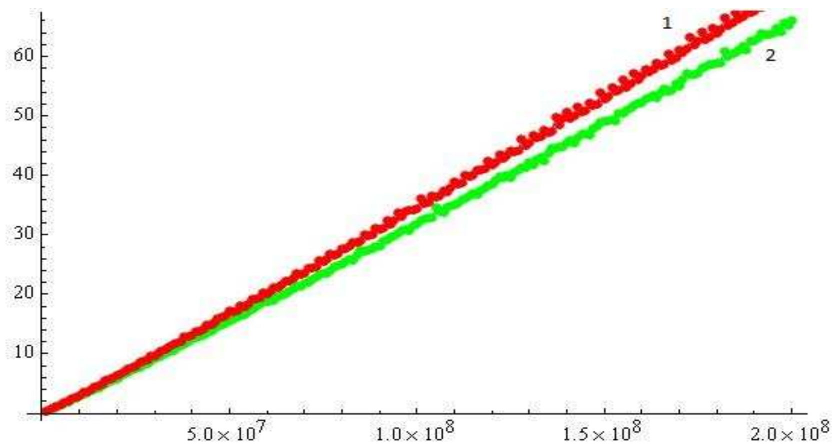

Figure 2: Algorithm 1 - Tembhurne-Sathe iterative (1 – red color), Algorithm 3 - Iliev–Kyurkchiev–Rahnev iterative (2 – green line)

REFERENCES

[1] A. Aho, J. Hopcroft, J. Ullman, *The Design and Analysis of Computer Algorithms*, 1st ed., Addison-Wesley, Boston (1974).

[2] A. Aho, J. Ullman, J. Hopcroft, *Data Structures and Algorithms*, 1st ed., Addison-Wesley, Boston (1987).

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

127.

[4] A. Akritas, G. Malaschonok, P. Vigklas, On the Remainders Obtained in Finding the Greatest Common Divisor of Two Polynomials, *Serdica Journal of Computing*, **9** (2015), 123-138.

[5] M. Alsuwaiyel, *Algorithms: Design Techniques and Analysis*, Lecture Notes Series on Computing, revised ed., World Scientific Publishing Company, Hackensack (2016).

[6] L. Ammeraal, *Algorithms and Data Structures in C++*, John Wiley & Sons Inc., New York (1996).

[7] T. M. Apostol, *Introduction to Analytic Number Theory*, Springer-Verlag, New York (1976).

[8] S. Baase, A. Gelder, *Computer Algorithms, Introduction to Design and Analysis*, 3rd ed., Addison-Wesley, Boston (2000).

[9] G. Brassard, P. Bratley, *Fundamentals of Algorithmics*, international ed., Pearson, (2015).

[10] D. Bressoud, *Factorization and primality testing*, Springer Verlag, New York (1989).

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

[12] Th. Cormen, *Algorithms Unlocked*, MIT Press, Cambridge (2013).

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

[14] R. Crandall, C. Pomerance, *Prime Numbers: A Computational Perspective*, Springer-Verlag, New York (2005).

[15] J. D. Dixon, The number of steps in the Euclidean algorithm, *J. Number Theory*, **2** (1970), 414-422.

[16] A. Drozdek, *Data Structures and Algorithms in C++*, 4th ed., Cengage Learning, Boston (2013).

[17] J. Erickson, *Algorithms*, University of Illinois Press (2009).

[18] J. Gareth, J. Jones, *Elementary Number Theory*, Springer-Verlag, New York (1998).

[19] 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)

[20] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer, New York (1996).

[21] S. Goldman, K. Goldman, *A Practical Guide to Data Structures and Algorithms Using JAVA*, Chapman & Hall/CRC, Taylor & Francis Group, New York (2008).

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

[23] M. Goodrich, R. Tamassia, D. Mount, *Data Structures and Algorithms in C++*, 2nd ed., John Wiley & Sons Inc., New York (2011).

[24] R. Graham, D. Knuth, O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed., Addison-Wesley, Boston (1994).

[25] D. H. Greene, D. E. Knuth, *Mathematics for the Analysis of Algorithms*, 2nd ed., Birkhauser, Boston (1982).

[26] H. A. Heilbronn, On the average length of a class of finite continued fractions. In: *Number Theory and Analysis (Turan, P., ed.)*, 87-96, Plenum Press, New York (1969).

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

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

[35] 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)

[36] 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, October 10-12, (2018). (to appear)

[37] A. Iliev, N. Kyurkchiev, The faster Euclidean algorithm, *Collection of scientific works from conference, Pamporovo, Bulgaria*, November 28-30, (2018). (to appear)

[38] A. Iliev, N. Kyurkchiev, The faster extended Euclidean algorithm, *Collection of scientific works from conference, Pamporovo, Bulgaria*, November 28-30, (2018). (to appear)

[39] P. Kyurkchiev, V. Matanski, The faster Euclidean algorithm for computing polynomial multiplicative inverse, *Collection of scientific works from conference, Pamporovo, Bulgaria*, November 28-30, (2018). (to appear)

[40] V. Matanski, P. Kyurkchiev, The faster Lehmer's greatest common divisor algorithm, *Collection of scientific works from conference, Pamporovo, Bulgaria*, November 28-30, (2018). (to appear)

[41] 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.

[42] A. Iliev, N. Kyurkchiev, A. Rahnev, 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.

[43] 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.

[44] 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, May 12-13, (2009), 52-58 (in Bulgarian), http://sci-gems.math.bas.bg/jspui/handle/10525/1356

[45] E. Kaltofen, H. Rolletschek, Computing greatest common divisors and factorizations in quadratic number fields, *Math. Comp.*, **53** (1990), 697-720.

[46] J. Kleinberg, E. Tardos, *Algorithm Design*, Addison-Wesley, Boston (2006).

[47] D. E. Knuth, Evaluation of Porter's constant, *Comp. Maths. Appls.*, **2** (1976), 137-139.

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

[49] Hr. Krushkov, *Programming in C#*, Koala press, Plovdiv (2017). (in Bulgarian)

[50] A. Levitin, *Introduction to the Design and Analysis of Algorithms*, 3rd ed., Pearson, Boston (2011).

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

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

[53] G. H. Norton, A shift-remainder GCD algorithm. In: Applied Algebra. Algebraic Algorithms and Error Correcting Codes (Huguet, L., Poli, A., eds.), *Springer LNCS*, **356** (1989), 350-356.

[54] G. H. Norton, On the Asymptotic Analysis of the Euclidean Algorithm, *J. Symbolic Computation*, **10** (1990), 53-58.

[55] J. W. Porter, On a theorem of Heilbronn, *Mathematika*, **22** (1975), 20-28.

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

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

[58] H. Rolletschek, On the number of divisions of the Euclidean algorithm applied to Gaussian integers, *J. Symbolic Computation*, **2** (1986), 261-291.

[59] H. Rolletschek, Shortest division chains in imaginary quadratic number fields. In: Symbolic and Algebraic Computation (Gianni, P., ed.), *Springer LNCS* **358** (1990), 231-243.

[60] D. Schmidt, *Euclid's GCD Algorithm* (2014).

[61] R. Sedgewick, K. Wayne, *Algorithms*, 4th ed., Addison-Wesley, Boston (2011).

[62] S. Skiena, *The Algorithm Design Manual*, 2nd ed., Springer, New York (2008).

[63] A. Stepanov, *Notes on Programming* (2007).

[64] E. Weisstein, *CRC Concise Encyclopedia of Mathematics*, Chapman & Hall/CRC, New York (2003).

[65] J. Stein, Computational problems associated with Racah algebra, *Journal of Computational Physics*, **1** (1967), 397-405.

[66] V. Harris, An algorithm for finding the greatest common divisor, *Fibonacci Quarterly*, **8** (1970), 102-103.

[67] T. Moore, On the Least Absolute Remainder Euclidean Algorithm, *Fibonacci Quarterly*, **30** (1992), 161-165.

[68] J. Tembhurne, S. Sathe, New Modified Euclidean and Binary Greatest Common Divisor Algorithm, *IETE Journal of Research*, **62**, No. 6 (2016), 852-858.