International Journal of Differential Equations and Applications
Volume **23**, No. 1 (2024), pages: 29-40
ISSN (Print): 1311-2872; ISSN (Online): 1314-6084;
url: https://www.ijdea.eu

AN EFFECTIVE REALIZATION OF DAYKIN-HARRIS ALGORITHM

Anton Iliev[1,2], Nikolay Kyurkchiev[1,2],
Asen Rahnev[1] and Svetoslav Enkov[1]

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

[2]Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8, 1113 Sofia, BULGARIA

ABSTRACT: In this paper we will give how it can be constructed optimized version of Daykin-Harris algorithm [1], [2] for computing greatest common divisor of two natural numbers. This algorithm can be useful for long numbers because of using mainly "subtraction" and "addition" operations.

AMS Subject Classification: 11A05, 68W01
Key Words: Euclidean algorithm, Harris algorithm, Daykin-Harris algorithm, medium numbers, long numbers

Received: April 14, 2024 **Revised:** May 20, 2024
Published: May 24, 2024 **doi:** 10.12732/ijdea.v23i1.3
Academic Publications, Ltd. https://acadpubl.eu

1. INTRODUCTION

The task of computation of greatest common divisor (g.c.d.) for natural numbers a and b can be used in many other tasks, see [3], [12]–[48]. In the last several years we develop many new algorithms for this topic [12]–[27].

Chronologically firstly Daykin [1] gives its algorithm. Several years later Harris [2] receive the same results using other more simpler iterative process. So this is the reason that we call the obtained optimized calculation way - effective Daykin-Harris algorithm.

We receive some speeding benefits by dividing the algorithm into two branches $a > b$ and $a \leq b$, also the $\max\{a, b\}$ in respectively computational branch is used in place of 10^k which is in originally work [1] as well as is more precise than this which is mention in [2].

Methodological remarks and estimations of numbers of operations for algorithms of Euclidean type are given in [5]-[11], [28]-[48].

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.

2. MAIN RESULTS

Our aims are to optimize both Harris [4] and Daykin-Harris [1], [2] algorithms which can be useful in their application to medium and long numbers, respectively.

2.1. A NOTE ON HARRIS ALGORITHM

Firstly here we will give new optimized version (named Algorithm 1. in our present paper) of other Harris algorithm [4], which is quite appropriate for medium numbers.

Algorithm 1.

```
int g = 0;
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;
while ((v & 1) == 0) v >>= 1;
if (u > v) do
{
u %= v;
if (u < 1) { gcd = v << g; break; }
if ((u & 1) == 0)
{
do u >>= 1; while ((u & 1) == 0);
```

```
if (u == 1) { gcd = u << g; break; }
}
else
{
u = v - u;
do u >>= 1; while ((u & 1) == 0);
if (u == 1) { gcd = u << g; break; }
}
v %= u;
if (v < 1) { gcd = u << g; break; }
if ((v & 1) == 0)
{
do v >>= 1; while ((v & 1) == 0);
if (v == 1) { gcd = v << g; break; }
}
else
{
v = u - v;
do v >>= 1; while ((v & 1) == 0);
if (v == 1) { gcd = v << g; break; }
}
} while (true);
else do
{
v %= u;
if (v < 1) { gcd = u << g; break; }
if ((v & 1) == 0)
{
do v >>= 1; while ((v & 1) == 0);
if (v == 1) { gcd = v << g; break; }
}
else
{
v = u - v;
do v >>= 1; while ((v & 1) == 0);
if (v == 1) { gcd = v << g; break; }
```

```
}
u %= v;
if (u < 1) { gcd = v << g; break; }
if ((u & 1) == 0)
{
do u >>= 1; while ((u & 1) == 0);
if (u == 1) { gcd = u << g; break; }
}
else
{
u = v - u;
do u >>= 1; while ((u & 1) == 0);
if (u == 1) { gcd = u << g; break; }
}
} while (true);
```

as well as its new optimized recursive version

Algorithm 2.

```
static long Euclid(long u, long v, int g)
{
if (u > v)
{
u %= v;
if (u < 1) return v << g;
if ((u & 1) == 0)
return Euclid(u >> 1, v, g);
else
{
if (u == 1) return u << g;
u = v - u;
if ((u & 1) == 0)
return Euclid(u >> 1, v, g);
}
}
else
```

```
{
v %= u;
if (v < 1) return u << g;
if ((v & 1) == 0)
return Euclid(u, v >> 1, g);
else
{
if (v == 1) return v << g;
v = u - v;
if ((v & 1) == 0)
return Euclid(u, v >> 1, g);
}
}
return Euclid(u, v, g);
}
```

The recursive function can be called by:

```
int g = 0;
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;
while ((v & 1) == 0) v >>= 1;
gcd = Euclid(u, v, g);
```

2.2. A NOTE ON DAYKIN-HARRIS ALGORITHM

We propose the following effective Daykin-Harris algorithm

Algorithm 3.

```
if (a > b)
{
```

```
z = a;
while ((a -= b) > b);
a1 = z - b;
if (a + a1 < z) do
{
do a1 += a; while (a + a1 < z);
if (a + a1 > z) do a += a1 - z; while (a + a1 > z);
} while (a + a1 != z);
else if (a + a1 > z) do
{
do a += a1 - z; while (a + a1 > z);
if (a + a1 < z) do a1 += a; while (a + a1 < z);
} while (a + a1 != z);
gcd = a;
}
else if (b > a)
{
z = b;
while ((b -= a) > a);
b1 = z - a;
if (b + b1 < z) do
{
do b1 += b; while (b + b1 < z);
if (b + b1 > z) do b += b1 - z; while (b + b1 > z);
} while (b + b1 != z);
else if (b + b1 > z)
{
do b += b1 - z; while (b + b1 > z);
if (b + b1 < z) do b1 += b; while (b + b1 < z);
} while (b + b1 != z);
gcd = b;
}
else gcd = a;
```

and we will compare the speed of Algorithm 3. to Euclidean algorithm [28] (Algorithm 4.).

Algorithm 4.

```
while (a != b)
if (a > b) a -= b; else b -= a;
gcd = a;
```

3. NUMERICAL EXAMPLE

For testing purposes of Algorithms 1., 2., 3. and 4. we will use the following main function:

```
long a, b, gcd, d1 = 0, u, v, a1, b1, z;
for (int i = 1; i < 100000001; i++) { a = i; b = 200000002 - i;
//here are placed the source code of algorithms 1., 2., 3. and
//calling of recursive algorithm 2.
d1 += gcd;
}
Console.WriteLine(d1);
```

CPU time results are:

CPU time of Algorithm 1. is: **27.596 seconds.**

CPU time of Algorithm 2. is: **51.779 seconds.**

CPU time of Algorithm 3. is: **78.450 seconds.**

CPU time of Algorithm 4. is: **91.054 seconds.**

It can be concluded that Algorithms 1. and 2. are much faster than classical Harris algorithm [4], which for the same numbers give **31.620 seconds** and **68.119 seconds** for the iterative and recursive realizations respectively.

It can be seen the computational speed benefits of Algorithm 3. in comparison to Algorithm 4.

4. CONCLUSION

Optimizations of classical algorithms is necessary to produce more computationally effective ways of solving the task of computing greatest common divisor of two natural numbers. The numbers can be divided into three categories with respect to their length - short, medium and long. In this paper we give computational processes for effectively computations for medium and long numbers. The question of existence of more efficient algorithms that use mainly "subtraction" and "addition" operations only remains still open.

ACKNOWLEDGEMENT

This study is financed by the project No FP23-FMI-002 "Intelligent software tools and applications in research in Mathematics, Informatics, and Pedagogy of Education" of the Scientific Fund of the Paisii Hilendarski University of Plovdiv, Bulgaria.

REFERENCES

[1] D. Daykin, An Addition Algorithm for Greatest Common Divisor, *Fibonacci Quarterly*, 8, No. 3 (1970), 347–349.

[2] V. Harris, On Daykin's algorithm for finding the g.c.d., *Fibonacci Quarterly*, 12 (1974), 80.

[3] E. Bach, J. Shallit, Algorithmic Number Theory, Vol. 1: Efficient Algorithms, MA: MIT Press, Cambridge (1996).

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

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

[6] 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)

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

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

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

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

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

[22] 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)

[23] 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.

[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, V. Kyurkchiev, New Extended Based on Generalization of Harris Algorithm, *Communications in Applied Analysis*, **26** No. 1 (2022), 61–73.

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

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

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

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

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

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

[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*, **7** No. 6 (2019), 10–18.

[35] J. L. 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.

[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*, **29**, (2021), 321–333.

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

[38] P. Thapar, U. Batra, Implementation of Elliptical Curve Cryptography Based Diffie-Hellman Key Exchange Mechanism in Contiki Operating System for Internet of Things, *International Journal of Electrical and Electronics Research (IJEER)*, **10** No. 2 (2022), 335–340.

[39] N. Fatma, M. R. Hassan, D. Akhtar, J. K. M. S. Zaman, Diminution of Extended Euclidean Algorithm for Finding Multiplicative Inverse in Galois Field, *Science and Engineering Journal*, **11** No. 1 (2023), 1222–1240.

[40] H. Qausar, M. Absa, A. T. Hidayat, Z. Mujtahid, Penerapan Pecahan Bersambung Dalam Melakukan Aproksimasi Bilangan Irasional Menuju Bilangan Rasional, *Jurnal Ilmiah Matematika Realistik (JI-MR)*, **4**, No. 1 (2023), 48–57.

[41] J. L. Butar-Butar, Y. B. P. Siringoringo, Kode Siklik Berulang Dari Kode Linear Fp Atas Lapangan Hingga F Pl Dengan L Bilangan Prima Tertentu, *Barekeng: J. Il. Mat. & Ter.*, **15**, No. 02 (2021), 231–240.

[42] V. Matanski, An Efficient Binary Algorithm for Solving Equation $GCD * 2^{|J-K|} = X * A0 + Y * B0$, Proceedings of Anniversary International Scientific Conference "Computer Technologies and Applications", 15-17 September 2021, Pamporovo, Bulgaria, *Plovdiv University Press*, 79–86, ISBN: 978-619-202-702-5.

[43] H. Gyulyustan, A Note on Euclidean Sequencing Algorithm, Proceedings of the Scientific Conference "Innovative ICT for Digital Research Space in Mathematics, Informatics and Educational Pedagogy", Pamporovo, 7-8.11.2019, *Plovdiv University Press*, (2020), 57–63, ISBN 978-619-202-572-4.

[44] P. Kyurkchiev, V. Matanski, The Faster Euclidean Algorithm for Computing Polynomial Multiplicative Inverse, Proceedings of the Scientific Conference Innovative ICT in Research and Education: Mathematics, Informatics and Information Technologies, Pamporovo, 29-30 November 2018, (2019), 43–48, ISBN: 978-619-202-439-0.

[45] V. Matanski, P. Kyurkchiev, The Faster Lehmer's Greatest Common Divisor Algorithm, Proceedings of the Scientific Conference Innovative ICT in Research and Education: Mathematics, Informatics and Information Technologies, Pamporovo, 29-30 November 2018, (2019), 37–42, ISBN: 978-619-202-439-0.

[46] D. J. Zhu, P. M. May, B. W. E. Norris, Z. M. Aman, E. F. May, Cage-Specific Hydrate Equilibrium Electrolyte Model, *Energy & Fuels*, (2024), doi: 10.1021/acs.energyfuels.3c05110

[47] Z. Ibran, E. Aljatlawi, A. Awin, On Continued Fractions and Their Applications, *Journal of Applied Mathematics and Physics*, **10** (2022), 142–159.

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