# THE FASTER EUCLIDEAN ALGORITHM

**Anton Iliev, Nikolay Kyurkchiev**

**Abstract:** In our previous works [12]–[22] we give a possible way to optimize classical widespread realizations of Euclidean algorithm. These algorithms are faster about 10% and about 30% in iterative and recursive ways of implementation respectively because we reduce some operations. We will note that recursive realizations of these algorithms [12]–[22] consume only about 50% of operation memory which is necessary in realizations given by other authors [3]–[11], [23]–[31]. The calculation of the greatest common divisor is one of the most important tasks in Number Theory. When ask Google for “Euclidean algorithm” in September 2018 we receive more than 10 500 000 indexed pages. Using well-known common apparatus for analysis we formulate the theorem which guarantee correctness of new [12]–[22] described by us Euclid’s algorithm.

***Keywords:*** *greatest common divisor, Euclid’s algorithm, reduced memory usage*

## 1. Introduction

The Euclidean algorithms and their modifications [1]–[31] work correctly for every natural numbers $a>0$ and $b>0$. Without loosing of generality we will explore the case when $a>b$. Let $a_0=a$ and $a_1=b$ (analysis of the other case $a\le b$ is analogical because the only that we need is to swap $a$ and $b$).

## 2. Main results

Let us denote by $m(m\ge 1)$ the number of divisions in the Euclid’s algorithm.

If $m$ is even number the process [12]–[22] can be defined by the following way:

Дата на получаване: 03.12.2018 г.
Дата на рецензиране: 29.03.2019 г.
Дата на публикуване: 17.05.2019 г.

$$
\begin{aligned}
&\textit{First step}:\\
&a_0 = c_0 a_1 + a_2,\ a_1 = c_1 a_2 + a_3;\\
&\textit{Second step}:\\
&a_2 = c_2 a_3 + a_4,\ a_3 = c_3 a_4 + a_5;\\
&\ldots\\
&\lfloor (m+1)/2 \rfloor \textit{ step}:\\
&a_{m-2} = c_{m-2} a_{m-1} + a_m,\ a_{m-1} = c_{m-1} a_m + a_{m+1},
\end{aligned}
\tag{1}
$$

where $c_j = \lfloor a_j / a_{j+1} \rfloor,\ 0 \le j \le m-1.$

It is obvious that if $m$ is odd number then every "step" will take one-half of "some step" of (1) and eventually one-half from "next step" of (1) which is no problem for organization of computational process. This division in "steps" is only for easier explanation of main idea.

Without loosing of generality we will explore the case when $m$ is an even number (the case when $m$ is an odd number is analogical as we already noted).

The process (1) will guarantee that we will keep correct order i.e. $(a_0 \ge a_1) > (a_2 > a_3) > \ldots.$

Let us denote the greatest common divisor of $a$ and $b$ by $g = gcd(a,b)$.

Here we will set that the faster Euclidean algorithm [12]–[22] is based on the following equivalence:

$$
\begin{aligned}
&(g \mid a) \text{ and } (g \mid b) \Leftrightarrow\\
&\left( g \mid \underbrace{a \bmod b}_{p}, p \ne 0, \right) \text{ and } \left( g \mid \underbrace{b \bmod p}_{q}, q \ne 0 \right).
\end{aligned}
\tag{2}
$$

We will point out that if $p = 0$ or $q = 0$ at the right hand side of equivalence (2) then $gcd(a,b) = b$ or $gcd(a,b) = p$ respectively.

So, $gcd(a,b) = gcd(a_2, a_3)$.

We will prove that $a_{m+1} = 0$.

Simple can be seen that $c_{m-1} \ge 2$ when $m > 1$.

Now we will prove that $a_m = g = gcd(a,b)$.

Using (1) on first step we receive $a_1 > a_0 \bmod a_1$, $a_2 = a_0 \bmod a_1$ and $a_2 > a_1 \bmod a_2, a_3 = a_1 \bmod a_2$ and we can conclude that $gcd(a_0, a_1) = gcd(a_2, a_3)$. On the second step: $a_3 > a_2 \bmod a_3$, $a_4 = a_2 \bmod a_3$ and $a_4 > a_3 \bmod a_4$,

$a_5 = a_3 \bmod a_4$ and we obtain that $gcd(a_2, a_3) = gcd(a_4, a_5)$. This process has limited number of divisions because $(a_0 \geq a_1) > (a_2 > a_3) > \ldots$ and on step number $\lfloor (m+1)/2 \rfloor$ for some $m$ we will receive $a_{m+1} = 0$.

These calculations can be written in algorithmic form:

```
Euclid(a,b)
        p = a mod b
      if (p = 0)
          return (b)
        q = b mod p
      if (q = 0)
          return (p)
      return Euclid(p,q)
```

The calling of algorithm Euclid$(a,b)$ is:

```
if (a > b)
        Euclid(a,b)
    else
        Euclid(b,a)
```

So we have proved the following:

**Theorem**. The Euclid's algorithm for natural numbers $a$ and $b$ gives as a result $gcd(a,b)$.

## Acknowledgments

This work has been supported by the project FP17-FMI008 of Department for Scientific Research, Paisii Hilendarski University of Plovdiv.

## References

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

[2] 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.

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

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

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

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

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

[8] 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)

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

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

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

[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. 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.

[14] 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.

[15] 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.

[16] 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(3) (2018), 31–34.

[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(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, *80th Anniversary of the birth of Prof. Donald Knuth*, Biomath Communications, 5 (2018), 7 pp.

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

[21] A. Iliev, N. Kyurkchiev, New Organizing of the Euclid's Algorithm and one of its Applications to the Continued Fractions, *Proc. of Scientific Conference "Mathematics. Informatics. Information Technologies. Application in Education"*, Pamporovo, Bulgaria, October 10–12, (2018). (to appear)

[22] A. Iliev, N. Kyurkchiev, The faster extended Euclidean algorithm, Proc. of *Scientific Conference "Innovative ICT in Research and Education: Mathematics, Informatics and Information Technologies", Pamporovo, Bulgaria*, November 29–30, 21–26, 2018.

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

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

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

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

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

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

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

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

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

Faculty of Mathematics and Informatics,

University of Plovdiv "Paisii Hilendarski",

24 Tzar Asen Str., 4000 Plovdiv, Bulgaria,

e-mails: aii@uni-plovdiv.bg, nkyurk@uni-plovdiv.bg

# ПО-БЪРЗИЯТ АЛГОРИТЪМ НА ЕВКЛИД

**Антон Илиев, Николай Кюркчиев**

**Резюме:** В наши предишни работи [12]–[22] даваме възможен път за оптимизиране на класически широкоразпространени реализации на алгоритъма на Евклид. Тези алгоритми са по-бързи с около 10% и с около 30% съответно при итеративна и рекурсивна реализация поради редуциране на някои операции. Ще отбележим, че рекурсивните реализации на тези алгоритми [12]–[22] консумират само около 50% от оперативната памет нужна на реализациите дадени от други автори [3]–[11], [23]–[31]. Изчисляването на най-големия общ делител е една от най-важните задачи в Теория на числата. При заявка към Google за “Euclidean algorithm” през септември 2018 г. получаваме повече от 10 500 000 индексирани страници. Използвайки добре известния общ апарат за анализ формулираме теоремата, която гарантира коректността на новия [12]–[22] описан от нас алгоритъм на Евклид.