International Journal of Pure and Applied Mathematics
Volume 117 No. 4 2017, 603-608
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
doi: 10.12732/ijpam.v117i4.5

A NOTE ON KNUTH'S IMPLEMENTATION OF EUCLID'S GREATEST COMMON DIVISOR ALGORITHM

Anton Iliev[1] §, Nikolay Kyurkchiev[2]

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

Abstract: In this note we give new and faster natural realization of Euclid's Greatest Common Divisor (GCD) algorithm. The reason of interest in this topic is widely application of this algorithm in various mathematics and computer science topics [13]. Particularly via Google you can see that there are more than 400 000 pages indexed for keyword 'greatest common divisor'. In our implementation we reduce the number of iterations and now they are 50% of Knuth's realization of Euclid's GCD. For all algorithms we have use the implementations in Visual C# 2017 programming environment.

To the bright memory of Prof. Iliya Iliev

AMS Subject Classification: 11A05, 68W01
Key Words: greatest common divisor, Euclid's algorithm, Knuth's algorithm, reduced number of iterations

1. Introduction

In all implementations we will use as comment in example a = 420748418; b = 9659595. All algorithms work correctly for every a>0 and b>0. We will mention that searching of new modifications of classical algorithms is serious task see for example our previous work on number of primes [12].

In his book Knuth [13] proposed the following iteration process:

Received: March 11, 2017
Revised: August 22, 2017
Published: January 23, 2018

§Correspondence author

Algorithm 1.

long a, b, ob, gcd; //a = 420748418; b = 9659595;

while (b > 0) { ob = b; b = a % b; a = ob; }

gcd = a;

which is the most commonly used and can be seen in many sources and books [3]–[11], [13]–[20].

The following algorithm is given by Schmidt [18]:
Algorithm 2.

long a, b, gcd; //a = 420748418; b = 9659595;

while (a > 0 && b > 0) if (a > b) a %= b; else b %= a;

gcd = a + b;

The next implementation is given by Stepanov [20]:
Algorithm 3.

long a, b, gcd; //a = 420748418; b = 9659595;

while (true) { if (b < 1) { gcd = a; break; }

a %= b; if (a < 1) { gcd = b; break; } b %= a; }

2. Main Results

Now we set the task to optimize all implementations of Euclid's algorithm. 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 in programming environment (see Fig. 1).

Figure 1: Visual C# 2017.

We propose the following iteration process after numerous attempts to optimize the Algorithms 1-3.

Algorithm 4.

```
long a, b, gcd;
//a = 420748418; b = 9659595;
    if (a > b) do { a %= b; if (a < 1) { gcd = b; break; }
    b %= a; if (b < 1) { gcd = a; break; } } while (true);
    else do { b %= a; if (b < 1) { gcd = a; break; }
    a %= b; if (a < 1) { gcd = b; break; } } while (true);
```

Numerical experiments.

Part 1. We will use the following task:

```
long d;
d = 0;
for (int i = 1; i < 1000000001; i++) { b = i; a = 2000000002 - i;
//here is the source code of every one of Algorithms 1-4
d += gcd; }
Console.WriteLine(d);
```

Results from Algorithms 1-4 (see Fig. 2 - Fig. 5).

Figure 2: Algorithm 1. d = 38332157136, time: 4 min. 10.016 sec.

Figure 3: Algorithm 2. d = 38332157136, time: 4 min. 15.994 sec.

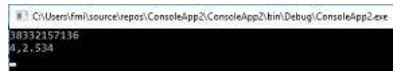

Figure 4: Algorithm 3. d = 38332157136, time: 4 min. 2.534 sec.

Figure 5: Algorithm 4. d = 38332157136, time: 3 min. 58.914 sec.

Figure 6: Algorithm 1. d = 38332157136, time: 4 min. 31.401 sec.

Figure 7: Algorithm 2. d = 38332157136, time: 4 min. 17.606 sec.

Figure 8: Algorithm 3. d = 38332157136, time: 4 min. 17.366 sec.

Figure 9: Algorithm 4. d = 38332157136, time: 4 min. 4.882 sec.

Part 2. We will use the following task where we swapped the values of 'a' and 'b' from Part 1:

```
long d;
d = 0;
for (int i = 1; i < 1000000001; i++) { a = i; b = 2000000002 - i;
//here is the source code of every one of Algorithms 1-4
d += gcd; }
Console.WriteLine(d);
```

Results from Algorithms 1-4 (see Fig. 6 - Fig. 9).

Part 3. Average time of performance
EN = (Part 1.Algorithm$N$ + Part 2.Algorithm$N$) / 2,
where $N = 1$ to 4 denotes using of Algorithms 1 to 4.
E1 = 4 min. 20.713 sec.; E2 = 4 min. 16.800 sec.;
E3 = 4 min. 9.950 sec.; E4 = 4 min. 1.898 sec.
So you can see that our new Algorithm 4 is faster than all others. This modification can be used for polynomial factorization [1], [5] and [2].

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. 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://scigems.math.bas.bg/jspui/handle/10525/1356

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

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

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

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

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

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

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

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