International Journal of Pure and Applied Mathematics
Volume 118 No. 1 2018, 31-37
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
doi: 10.12732/ijpam.v118i1.3

ijpam.eu

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

Anton Iliev[1 §], Nikolay Kyurkchiev[2], Angel Golev[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 give new and faster natural realization of Extended Euclidean Greatest Common Divisor (EEGCD) algorithm. The motivation of this work is that this algorithm is used in numerous scientific fields [36], [24]. Internet search engines show very high appearance of 'greatest common divisor'. In our implementation we reduce the number of iterations and now they are 50% of Knuth's realization of EEGCD. For all algorithms we have use the implementations in Visual C# 2017 programming environment.

This paper is dedicated to Prof. Donald Knuth
on occasion on his 80th anniversary

AMS Subject Classification: 11A05, 68W01

Key Words: greatest common divisor, extended Euclidean greatest common divisor, 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. Our work is natural continuation of ideas given in [21], [22]. In previous paper [21] we gave new natural algorithm for GCD. Here we will write it recursively in this elegant manner:

Received: April 10, 2017
Revised: January 6, 2018
Published: February 14, 2018

§Correspondence author

```
static long Euclid(long a, long b)
{ long r = a % b; if (r < 1) return b;
long u = b % r; if (u < 1) return r;
return Euclid(r, u); }
```

This algorithm is also about 30% faster than recursive implementation of Knuth's algorithm [24]:

```
static long Euclid(long a, long b)
{ if (b < 1) return a; long r = a % b;
return Euclid(b, r); }
```

which is given as Algorithm 1 in [21].
We can call every of both functions via the following operator:
if (a > b) gcd = Euclid(a, b); else gcd = Euclid(b, a);

In his book Knuth [24] proposed the following iteration process:

Algorithm 1.

```
x1 = 1; x2 = 0; //ao = 420748418; bo = 9659595;
while (b > 0) { q = a / b; r = a % b; a = b; b = r;
t = x2; x2 = x1 - q * x2; x1 = t; }
eegcd = a; x = x1; y = (a - x * ao) / bo;
```

which is widely spread via many sources and books [1]-[20] and [23]-[36].

Note that in all algorithms 'ao' and 'bo' are initial values of 'a' and 'b' respectively.

Recursive implementation of Algorithm 1. is:

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; }
```

and its calling:
if (a > b) eegcd = Euclid(a, b, ref x, ref y);

Figure 1: Visual C# 2017.

```
else eegcd = Euclid(b, a, ref x, ref y);
```

2. Main Results

Now we set the task to optimize Knuth's implementations of EEGCD 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 with the following programing environment (see Figure 1).

We propose the following iteration process.

Algorithm 3.

```
//ao = 420748418; bo = 9659595;
if (a > b) { x1 = 1; x2 = 0;
do { q = a / b; a %= b; t = x2; x2 = x1 - q * x2; x1 = t;
if (a < 1) { eegcd = b; x = x1; y = (b - x * ao) / bo; break; }
q = b / a; b %= a; t = x2; x2 = x1 - q * x2; x1 = t;
if (b < 1) { eegcd = a; x = x1; y = (a - x * ao) / bo; 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) { eegcd = a; x = x1; y = (a - x * ao) / bo; break; }
q = a / b; a %= b; t = x2; x2 = x1 - q * x2; x1 = t;
if (a < 1) { eegcd = b; x = x1; y = (b - x * ao) / bo; break; } } while (true); }
```

Recursive variation of Algorithm 3. is the following:

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; }
```

and the calling is:

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

Numerical experiments.

Part 1. We will use the following task:

```
long a, b, x1, x2, x, y, q, r, t, ao, bo, eegcd;
//a = 420748418; b = 9659595;
long d;
d = 0;
for (int i = 1; i < 100000001; i++)
{ b = i; a = 200000002 - i; bo = i; ao = 200000002 - i;
//here is the source code or callings when it is recursive
//implemented every one of Algorithms 1-4
d += eegcd; }
Console.WriteLine(d);
```

Results of Part 1.: Time of Algorithm 1: 32.473 sec.; Time of Algorithm 2: 61.090 sec.; Time of Algorithm 3: 31.922 sec.; Time of Algorithm 4: 45.607 sec.

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

```
long a, b, x1, x2, x, y, q, r, t, ao, bo, eegcd;
//a = 420748418; b = 9659595;
long d;
d = 0;
for (int i = 1; i < 100000001; i++)
```

```
{ a = i; b = 200000002 - i; ao = i; bo = 200000002 - i;
//here is the source code or callings when it is recursive
//implemented every one of Algorithms 1-4
d += eegcd; }
Console.WriteLine(d);
```

Results of Part 2.: Time of Algorithm 1: 35.065 sec.; Time of Algorithm 2: 59.604 sec.; Time of Algorithm 3: 31.438 sec.; Time of Algorithm 4: 45.579 sec.

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 = 33.769 sec.
E2 = 1 min. 0.247 sec.
E3 = 31.680 sec.
E4 = 45.593 sec.

So you can see that our new Algorithms 3 and 4 are faster than Algorithms 1 and 2 respectively.

Acknowledgments

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

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

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

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

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

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

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

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

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

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

[16] 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)

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

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

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

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

[21] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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