The Euclidean Algorithm

Deniz Gurel

December 5th 2014

Abstract

The greatest common divisor or greatest common factor of two positive integers is a familiar concept to most anyone who has made it through high school. Yet, while the concept is seemingly simple to understand, the method for computing the greatest common divisor has many complex applications. The method known as the Euclidean algorithm is a beatifully simplistic way to find the greatest common divisor of two arbitrary integers. Using only subtraction, multiplication, and inequalities, the Euclidean algorithm is one of the easiest numerical algorithms to understand, but throughout history it has proven essential in soliving difficult problems in both mathematics and computer science.

Contents

1 Introduction

The earliest printed instance of the Euclidean algorithm is in the greek mathematician Euclid's famous book titled *Euclid's Elements* written in 300 B.C. Euclid uses the familiar concept of integer distances to craft an abritrary way of finding the greatest common divisor of two integers that are not relatively prime.

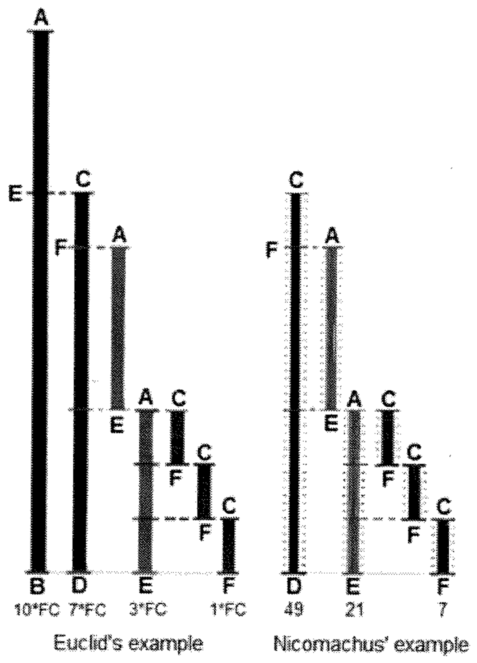

Figure 1: An illustrated version of Euclid's algorithm

Euclid's example above comes straight from proposition 2 from book VII of Euclid's Elements. Euclid basically states that in order find the gcd of two integers AB and CD such that $AB > CD$ one must start by finding the remainder of AB/CD. In the Euclidean example above this remainder is AE. The proccess is then repeated with CD and AE until the remainder is zero.

In Euclid's example the remainder of AE/FC is zero, thus $GCD(AB, CD) = FC$. Below is an application of the algorithm using 1071 and 462.

$$1071 = (2 \times 462) + 147 \tag{1}$$
$$462 = (3 \times 147) + 21 \tag{2}$$
$$147 = (7 \times 21) + 0 \tag{3}$$
$$GCD(1071, 462) = 21 \tag{4}$$

1.1 Prior Experience

While not taught outright the Euclidean algorithm has applications in many fields of mathematics and computer science. Since the Euclidean algorithm is relatively simple to grasp it is often used to demonstrate the idea of recursion for students in introductory level courses.

1.1.1 MAT 3110 Modern Algebra

The early sections discussed in modern algebra covered concepts such as relatively prime numbers and divisibility. A powerful proposition was indtroduced stating that two integers a and b are relatively prime (GCD(a,b) = 1) if and only if there are two integers m and n such that $ma + nb = 1$. This proposition allowed us to think about numbers in a sense that we generally were not used to. However this proposition was only useful if we knew two numbers were relatively prime, which is where Euclid's algorithm comes into the picture. Later on in the semester, we discussed the real-world applications of abstract algebra, specifically in the area of RSA encryption. The RSA algorithm uses many concepts from abstract algebra in order to encrypt and decrypt information. There is a crucial step in the RSA algorithm where the Euclidean algorithm can be applied. Without the ability to find multiplicative inverses in finite rings, the RSA algorithm would fail, luckily the Euclidean algorithm can be manipulated to find an efficient solution.

1.1.2 CS 2440 Computer Science II

In the second level of computer science study at Appalachian State we began a more rigorous approach to the problems that present themselves in computer programming. In computer science recursion is the concept where a function will call itself. Recursion became an area of focus throughout the course as it is generally one of the more difficult concepts to understand. As the semester went on I would search for examples to help me understand recursion, which is where I stumbled upon the Euclidean algorithm. The algorithm can be especially helpful for students because the recursion is naturally there, and it doesn't feel forced like many other examples in computer science classes.

1.1.3 CS 3460 Data Structures

Data Structures was a class that focused primarily on how data is manipulated using software. The course focused on the implementations of arrays, trees, and hash maps. In order to do well in the course we needed to complete long and complex programming assignments which required the use of many non-trivial algorithms and data structures. In a certain programming assignment I found that I needed to solve for the greatest common denominator of three different numbers, so I applied my knowledge of the Euclidean algorithm to help solve the problem. Below is the code written in C:

```c
int gcd(int a, int b, int c){
    //The gcd(a, b, c) is the same as the gcd(a, gcd(b, c)
    return euclid(a, euclid(b, c));
}

int euclid(int a, int b){

    if (b == 0){
        return  a;
    } else {
        //the '%' means modulus: a % b is
        //equivalent to the remainder of a / b
        return euclid(b, a % b);
    }
}
```

As you can see above the implementation of Euclid's algorithm is both elegant and simple, and it is relatively efficient in terms of processing speed. In the function named euclid you can see that if the parameter b is 0 then we return the parameter a. This is for the end case of the Euclidean algorithm where the remainder is zero. Since the remainder is always passed along as the second parameter we must check to see if the algorithm is finished. Variations of the function named euclid have been implemented in many different applications to solve a wide variety of software based problems.

2 Developments

Since the Euclidean algorithm has been around for a substantial period of time, there have been a large number of works applying the algorithm in one way or another. Because of the breadth of topics that utilize the Euclidean algorithm I will only cover a few important applications of the algorithm.

2.1 History

2.1.1 Earliest Work

Euclid's algorithm was first printed in the book *Euclid's Elements* around the year 300 B.C. Euclid discussed the algorithm in a geometrical using between named points to represent integers, as this was a simple way of showing remainders. The algorithm proved to be useful in the following chapters of *Euclid's Elements* since Euclid used the algorithm in many other corollaries throughout the text. Many historians believe that the algorithm was first introduced centuries before Euclid's life, since many of the examples in *Euclid's Elements* were merely compiled by Euclid and not discovered.[1]

2.1.2 19th Century

In the 1800s Carl Friedrich Gauss used the Euclidean algortithm in showing the unique factorization Gaussian integers. Peter Gustav Lejeune Dirichlet was one of the first to realize the importance of the algorithm to number theory, and he applied numerical theorems number sets where the Euclidean algorithm could also be applied. In the late 19th century Richard Dedekind expanded on Dirichlet and Gauss' work and proved Fermat's two square theorem using the unique factorization of gaussian integers. Dedekind also defined what is known as a Euclidean domain, a set of elements that form a communative ring under two general operations, where the Euclidean algorithm can be applied in a much more general sense.[2]

2.1.3 20th Century

The development of computers in the 1900s opened up a brand new avenue of applications for the Euclidean algorithm. In 1977, Ron Rivest, Adi Shamir and Leonard Adleman published a paper detailing how a public encryption key can be applied to a message that can only be decrypted by a private decryption key. The ecryption method became known as RSA encryption for the first letter of each author's name. A key piece of the encryption algorithm developed by Rivest, Shamir, and Adleman requires finding multiplicative inverses in large prime rings, which can easily be done using the euclidean algorithm.[3]
Below is an example of finding the multiplicative inverse of 20 in a ring with the operations multiplication and addition modulus 97.

We will start by applying the algorithm to (20, 97)

$$\begin{aligned}97 &= (4 \times 20) + 17\\20 &= (1 \times 17) + 3\\17 &= (5 \times 3) + 2\\3 &= (1 \times 2) + 1\end{aligned}$$

By rewriting the above equations so that the numbers to the right of the addition (17, 3, 2, and 1) are isolated we can substitute and solve for 1 as a sum of products.

$$\begin{aligned}
17 &= 97 - (4 \times 20) \\
3 &= 20 - 17 \\
&= 20 - (97 - (4 \times 20)) \\
&= -97 + (5 \times 20) \\
2 &= 17 - (5 \times 3) \\
&= (97 - (4 \times 20)) - (5 \times (-97 + (5 \times 20)) \\
&= (6 \times 97) - (29 \times 20) \\
1 &= 3 - 2 \\
&= (-97 + (5 \times 20)) - ((6 \times 97) - (29 \times 20)) \\
&= (-7 \times 97) + (34 \times 20)
\end{aligned}$$

We can clearly see that $(34 \times 20) = 1 + (7 \times 97)$. Since we are using a ring under addition/multiplication modulus 97, any multiple of 97 can effectively be ignored, thus we can see that 34 is the multiplicative inverse of 20 in a mod 97 ring. This extension of the Euclidean algorithm allows modern computers to find multiplicative inverses of large numbers in even larger prime rings in virtually no time.
The Euclidean algorithm has also proved useful in error correcting codes such as Reed-Solomon codes based on Galois fields. The algorithm proves useful with fields similarly to the method shown above. [4]

2.2 Recent Scholarly Research

The diversity of the Euclidean algorithm is evident in mathematical and computer science research in the present day. As Donald Knuth once said, "[The Euclidean algorithm] is the granddaddy of all algorithms, because it is the oldest nontrivial algorithm that has survived to the present day."[5] Current mathematical research involves generalizing the algorithm and applying it to varying concepts in number theory. Once specific example is M.R. Murty and Kathleen Petersen's paper on primitive roots and finite fields. They discussed how the Euclidean algorithm can be generalized to find primitive roots of finite number fields. Murty and Kathleen were able to determine various interesting results related to Galois fields and Euclidean domains.[6]

In fields relating to computation the current research is generally related to modifying the Euclidean algorithm as to increase performance. While the conventional Euclidean algorithm is efficient, it can still take longer than desired even on modern machines. When the algorithm is generalized to different numerical constructs the coded version ends up being more complex. By modifying the algorithm slightly for specific purposes researchers are able to increase the efficiency. Even though current computers are powerfull enough to execute the code with smaller numbers, algorithmic efficiency is essential when dealing with larger scale numbers.[7]

3 Acknowledgements and References

References

[1] Euclid. *Elements, Book VII.* Alexandria: 300 BCE

[2] Moon, Todd K.*Error correction coding: mathematical methods and algorithms.* Hoboken, New Jersey: John Wiley and Sons, 2005

[3] R.L. Rivest, A. Shamir, and L. Adleman *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems* Communications of the ACM 21,2 (Feb. 1978), 120–126.

[4] Greenfield, Stephen J. *Supplementary material for the lecture of Monday, July 12.* Available online: `http://www.math.rutgers.edu/~greenfie/gs2004/euclid.html` (accsessed October 2014)

[5] Knuth, Donald *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms.* 1981

[6] Murty, M. R. and Kathleen Peterson *The Euclidean algorithm for number fields and primitive roots.* Proceedings of the American Mathematical Society (v. 141/181-190, 2013)

[7] Roy, Marie-Francoise and Sidi Mohamed Sedjelmac *New fast Euclidean Algorithms.* Journal of Symbolic Computation (v. 50/208226, 2013)

Author's e-mail address: gureldm@gmail.com