Euclid's Algorithm/Extended Euclid's Algorithm and Applications

Grayson Goering, Jared Smerdell, Jacob Ewy, Dennon Parks

November 2020

1 Introduction

The field of number theory was seen as one of the less useful areas of math for much of its existence. Concerned with the relationships and patterns we find in the natural numbers and the integers, it was for a long time seen as mostly a beautiful area of mathematics that, while interesting, never gave us many important applications. Much of number theory is still just for beauty and fun in academia, but parts of it have come to be very useful to computer scientists. Euclid's algorithm (first published in Euclid's Elements) is one of the oldest algorithms that is still in use by computer scientists today, and has a variety of applications. These include but are not limited to security, computer graphics, neutron accelerator timing, musical rhythm generation. Euclid's algorithm is sometimes referred to as the grandfather of all algorithms since it is one of the oldest algorithms that is still used by computer scientists today and hasn't been refined to a better solution.

2 Problem Statement

Euclid's algorithm is used to find the greatest common divisor of 2 numbers. It is also the fastest computational technique to do so as finding factors of numbers is expensive. Below we will explain the steps of both the standard Euclid's algorithm, which finds some d where d is the GCD of 2 integers a and b, and the extended Euclid's algorithm which also finds some x, y such that $d = ax + by$. Therefore the input for both of our algorithms will be some non-negative integers a and b.

3 Applications

While Euclid's algorithm had only a few mathematical applications (involving factorization) for a long time, it is now used in a wide variety of application in computer science. The most common use for Euclid's algorithm and the

Extended Euclid's algorithm is in encryption. Some form of Euclid's algorithm is used in RSA encryption which is still a standard encryption used on smaller amounts of data, while AES is currently the most common encryption used on the internet today. Euclid's algorithm is also used in elliptic curve cryptography. The algorithms also have a number of applications that are related to the idea of finding the most even way to distribute data across a domain, which we can do using the Euclid's algorithm. These applications that use this even distribution technique include generating linear computer graphics, generating musical rhythms, spallation neutron source accelerators, Markoff numbers and two-distance sequences, and calculating leap years. The number of applications that use Euclid's algorithm and the Extended Euclid's algorithm is huge and the applications it is used for span a wide variety of fields including math, computer science, physics, etc.

4 Recent Work

4.1 Diophantine Equations

Problem: We need to develop an algorithm to generate the set of all solutions to a system of linear Diophantine equations with lower and upper bounds on the variables.

Reason for Caring: Using Euclid's Algorithm we can make use of the ability to parametrise the set of all solutions to a linear Diophantine equation in two variables with a single parameter.

Solution: The solution with return if there are any integral solutions or not and if the solutions numbers are prime and the greatest common divisor is 1 we can do even more with RSA encryption. Reason for Caring about Solution: If we have an integral solution this means that the Diophantine equation will not have an infinite amount of solutions. Otherwise, if we cant find any integral solutions the Diophantine equation has an infinite amount of answers. RSA: Using the integral solution to the Diophantine equation we can use the two prime numbers as our Keys in our RSA encryption. Then we can use these keys to give restricted access to certain files and parts of your system.

4.2 Generating Traditional Musical Rhythms

In 2005 Godfried Toussaint published a paper describing a method for using the Euclidean algorithm to generate traditional African bell rhythms. The Euclidean algorithm is used to help create patterns that are as evenly distributed as possible. The idea of creating evenly distributed patterns is also useful in several other applications including generating linear computer graphics, spallation neutron source (SNS) accelerators in nuclear physics, Sturmian words and string theory, Markoff numbers and two-distance sequences in number theory, and calculating leap years in calendar design, which just shows how useful just one use of the Euclidean applications is.

The idea of representing music mathematically is not a new one. Pythagoras expressed musical harmony as ratios of small integers, and many other mathematicians have done different kinds of work to model music. Most of this work however has been to model ideas of pitch, harmony, scales, and tuning, much less work has been done to model ideas about rhythm in a mathematical perspective more than what is already established music theory.

Godfried Toussaint's paper describes Euclidean rhythms, denoted E(x, y) where $x \leq y$. X represents the number of pulses in the rhythm that have sound and y represents the total number of of pulses (in a measure for example). The goal of a Euclidean rhythm algorithm is to create a rhythm where the pulses with sound are evenly distributed. For example E(3, 8) will render the list of pulses [1,1,1,0,0,0,0,0] and the Euclidean rhythm will be [1,0,0,1,0,0,1,0]. The Euclidean rhythm E(3,8) is actually a common rhythm in Cuban music known as a tresillo.

The Euclidean rhythms where $x|y$ will be in the form where there is an active pulse followed by $(y/x) - 1$ blank pulses. The more interesting case is when x/y has a remainder. In this case the Euclidean algorithm is used to find the most even distribution of x over y, which is by definition the Euclidean rhythm.

Godfried Toussaint describes a wide variety of different rhythm types that are used in commonly in different types of music across the world. He defines types of rhythms where x is prime, y is prime, x and y are prime, and shows how these different classes of Euclidean rhythms are more common in different parts of the world. The paper uses the Euclidean algorithm to analyse a very different idea than many of the other ways it is commonly used. The paper also connects this use of Euclid's algorithm to several other uses that seem more applicable to math and computer science.

4.3 Elliptic Curve Cryptography

Elliptic curve cryptography is currently one of the most secure methods of encryption that we know of. It can offer the same levels of security that we get from RSA (which was the most secure encryption standard from 1977-1997 when it was unclassified), but used much shorter key lengths. However Elliptic Curve Cryptography can also be very slow because the process is dependent on building an efficient system to do EC point multiplication.

Apostolos P. Fournaris and O. Koufopavlou pulished a paper in 2007 that uses a modified version of the Extended Euclid's algorithm to perform eliptic curve cryptography. More specifically they used the Extended Euclidean algorithm to build a system that can perform all of the GF(2^k) operations (addition/subtraction, multiplication/inversion). The Extended Euclid's algorithm can be used to solve all of the GF(2^k) operations but it doesn't return results in a constant number of iterations so the authors develop a modified version of the algorithm that is suitable for the task.

In conclusion the authors use a modified version of the Extended Euclid's algorithm (that can be implemented well in hardware) that can be used as a GF(2^k) arithmetic unit. This logic unit has a desirable time complexity while

also being able to perform all of the GF(2^k) that are required to perform elliptic curve cryptography. This then gives us the ability to build this system into machines architectures so that they can efficiently do all of the operations necessary in elliptic curve cryptography. Thus the system can be used as an efficient way to do elliptic curve cryptography, which we know to be an improvement on RSA because it has shorter key length.

5 Algorithm Solution

5.1 Euclid's Algorithm Example

5.1.1 Pseudo code

```
1    Euclid(a,b):
2        if b == 0:
3            return a
4        else:
5            return Euclid(b, amod(b))
```

5.1.2 General Process

Euclid's algorithms depends on the following theorem (later refered to as thm 1):
$gcd(a, b) = gcd(b, amod(b))$

-We want to find $gcd(a, b)$, but it is much easier to find $gcd(x, 0)$ for some integer x. This is because we know $gcd(x, 0) = x \ \forall \ x \in Z$, since 0 is divided by all integers and the greatest divisor of x is always x.
-Then since it is easy to find $gcd(x, 0) = x \ \forall \ x \in Z$, we want to reduce $gcd(a, b)$ to the form $gcd(x, 0)$.
-We can reduce $gcd(a, b)$ to the form $gcd(x, 0)$ because of thm 1. As we recursively call Euclid(b, a*mod(b)) on line 5, each recursive call will be equal to the original $gcd(a, b)$ we wanted to find. Each time we make the recursive call it calculates $amod(b)$ which is always less than a. Thus we will either get to some call where $amod(b) = 0$, or $0 < amod(b) < a$. If $amod(b) = 0$ then we will make the call Euclid(b, 0) and enter the if statement on line 2 and return b since $gcd(b, 0) = a$.
-The other case is that we keep making recursive calls where $amod(b) < a$ until we make the call Euclid(x, 1) for some x. In this case $gcd(x, 1) = 1$ because the greatest divisor of 1 is 1 and 1 divides all numbers. Thus in this case $gcd(a, b) = 1$ since $gcd(a, b) = gcd(b, amod(b))$.
-Thus Euclid's algorithm has 2 cases.
1. $gcd(a, b) = gcd(x, 0) = x$ for some $x \in Z$
2. $gcd(a.b) = gcd(x, 1) = 1$ for some $x \in Z$

5.1.3 Specific Example 1

Example input: a = (48, 36)

Steps of execution:

1. On line 1 call Euclid(48, 36).
-This will enter the else on line 4 because 36 != 0, and return Euclid(36, 48 mod 36), note that $gcd(48, 36) = gcd(36, 12)$
2. On line 5 call Euclid (36, 12)
-This will enter the else on line 4 because 12 != 0, and return Euclid(12, 36 mod 12), note that $gcd(48, 36) = gcd(36, 12 = gcd(12, 0)$
3. On line 5 call Euclid(12, 0)
-This will enter the if on line 2 since 0 = 0. Notice that since any number divides 0 and the greatest divisor of 12 is 12 we know that $gcd(12, 0) = 12$. This is confirmed by the algorithm which will return 12 on line 3 because of this observation.
4. Thus our algorithm will return 12 which is indeed the greatest common divisor of 48 and 36. Note $gcd(48, 36) = gcd(36, 12 = gcd(12, 0) = 12$

5.2 Extended Euclid's Algorithm Example

5.2.1 Pseudo code

```
1: procedure EXTENDED-EUCLID(a, b)
2:     if b == 0 then
3:         return (a, 1, 0)
4:     else(d', x', y') = Extended-Euclid(b, a mod b)
5:         (d, x, y) = (d', y', x' − (⌊a/b⌋ y'))
6:         return (d, x, y)
7:     end if
8: end procedure
```

5.2.2 Specific Example

Example input: Extended-Euclid(48, 36)

Steps of execution:

1. The function checks if $b == 0$ (line 2), 36! = 0 so the function enters the else statement. The purpose of this check is to terminate the recursion once the greatest common divisor has been found.
2. The function recursively calls itself with (d', x', y') = Extended-Euclid(36, 48 mod 36) = Extended-Euclid(36, 12) (line 4). Each recursive call is calculating

the $gcd(48, 36)$, same as the standard Euclid algorithm.

a	b	$\lfloor a/b \rfloor$	d	x	y
48	36	1	-	-	-
36	12	3	-	-	-

3. Once again the function checks if it has reached the gcd. $12! = 0$ (line 2), so it enters the else statement and calls itself again (line 4). The next call is (d', x', y') = Extended-Euclid(12, 36 mod 12) = Extended-Euclid(12, 0).
4. This time the termination case is met as $b = 0$ (line 2). The function now knows the $gcd(48, 36) = gcd(36, 12) = gcd(12, 0) = 12$.
5. The return value of the termination case is $(d', x', y') = (12, 1, 0)$ (line 3). Note that $d = d' = gcd(48, 36) = 12$ throughout the rest of the function.

a	b	$\lfloor a/b \rfloor$	d	x	y
48	36	1	-	-	-
36	12	3	-	-	-
12	0	-	12	1	0

6. Now that there is a set of values for (d', x', y') the recursive calls can begin unwinding. The a and b values are the values we passed to the current call; in this case $(a, b) = (32, 12)$.
7. The current call has been suspended at line 4 but now has all the information needed to evaluate the expression $(d, x, y) = (12, 0, 1 - (\lfloor 36/12 \rfloor * 0)) = (12, 0, 1)$ (line 5).
8. The function finishes by returning $(12, 0, 1) = (d', x', y')$ to the caller (line 6).

a	b	$\lfloor a/b \rfloor$	d	x	y
48	36	1	-	-	-
36	12	3	12	0	1
12	0	-	12	1	0

9. Once again, all the values are satisfied to evaluate the expression $(d, x, y) = (12, 1, 0 - (\lfloor 48/36 \rfloor * 1)) = (12, 0, -1)$ (line 5). The function finishes and returns the final value of $(d = 12, x = 1, y = -1)$.

a	b	$\lfloor a/b \rfloor$	d	x	y
48	36	1	12	1	-1
36	12	3	12	0	1
12	0	-	12	1	0

10. Now that the integer coefficients have been calculated as well as the $gcd(48, 36)$ the equation $d = gcd(a, b) = ax + by$ has been solved. Giving a final equation of $gcd(48, 36) = 12 = (1)48 + (-1)36$

5.3 Necesary Proofs

5.3.1 Theorem 31.5

Theorem 31.5: If $a|b$ and $b|a$ then $a = b$.

Proof:
Assume $a, b \geq 0$
If $a|b$ then either $a \leq b$ or $b = 0$
If $b|a$ then either $b \leq a$ or $a = 0$
Then if $a|b$ and $b|a$ we know $a \leq b \leq a$ which implies that $a = b$.

5.3.2 Theorem 31.3

Theorem 31.3: For any $a, b \in Z$, if $d|a$ and $d|b$ then $d|gcd(a, b)$

Proof:
By theorem 31.4 (below) $gcd(a, b) = ax + by$ for some $x, y \in Z$.
If $d|a$ then $a = dk$ for some $k \in Z$.
If $d|b$ then $a = dL$ for some $L \in Z$.
Then it follows $gcd(a, b) = ax + by = dkx + dLy$ for some $d, L \in Z$
It is obvious that $d|dkx + dLy$ so we know for any $a, b \in Z$, if $d|a$ and $d|b$ then $d|gcd(a, b)$.

5.3.3 Theorem 31.4

Theorem 31.4: If $d|a$ and $d|b$ then $d|(ax + by)$.

Proof:
$d|a$ implies $a = dk$ for some $k \in Z$
$d|b$ implies $b = dL$ for some $L \in Z$
Then it follows $ax + by = (dkx + dLy)$ so it is obvious $d|(ax + by)$.

5.3.4 Theorem 3.8

Theorem 3.8: $amodb = a - b\lfloor a/b \rfloor$.

Proof:
Assume $a \geq b$
Notice that $amodb$ is the remainder of the quotient a/b
Then it follows that $b * \lfloor a/b \rfloor$ is the non remainder portion of a/b since it is the number of times b can go evenly into a.
Then since $a \geq b$ $a - b\lfloor a/b \rfloor$ is the remainder portion of a/b.
Thus it follows that $amodb = a - b\lfloor a/b \rfloor$

5.3.5 Theorem 31.9

Theorem 31.9: If a, b are non-negative integers then $gcd(a, b) = gcd(b, amodb)$.
Note that this is the central proof needed to verify Euclid's algorithm since it recursively makes calls to Euclid(b, a mod b).

Proof:
Our goal will be to show that $gcd(a, b)|gcd(b, amodb)$ and $gcd(b, amodb)|gcd(a, b)$, since by thm 31.5 this will imply $gcd(a, b) = gcd(b, amodb)$.
First we will show $gcd(a, b)|gcd(b, amodb)$:
Let $d = gcd(a, b)$ then we know $d|a$ and $d|b$
By thm 3.8 we know $amodba - qb$, where $q = \lfloor a/b \rfloor$
Then it follows $amodb$ is a linear combination of a and b so $amodb = ax + by$
Then since we know $d|a$ and $d|b$ by thm 31.4 we know $d|amodb$
By thm 31.3 since $d|amodb$ and $d|b$ we know $d|gcd(b, amodb)$ so $gcd(a, b)|gcd(b, amodb)$
Now we need to show $gcd(b, amodb)|gcd(a, b)$
Let $d = gcd(b, amodb)$, then $d|b$ and $d|amodb$
Then $a = qb + (amodb)$ where $q = \lfloor a/b \rfloor$
Then it follows a is a linear combination of b and $amodb$
By thm 31.4 we know $d|a$
Since $d|b$ and $d|a$ we know $d|gcd(a, b)$ so $gcd(b, amodb)|gcd(a, b)$
Then we have $gcd(a, b)|gcd(b, amodb)$ and $gcd(b, amodb)|gcd(a, b)$ so by thm 31.5 we know $gcd(a, b) = gcd(b, amodb)$

5.4 Algorithm Proofs

5.4.1 Euclid's Algorithm Proof

Euclid's algorithm has 2 possible paths of execution:
Case 1: The algorithm makes a recursive call Euclid(b, a mod b) on line 5.
We know from thm 31.9 that gcd(a,b) = gcd(b, a mod b) so if this recursive call results in a solution, that solution is also the solution to the original call Euclid(a,b). This recursion tree can go down as many steps necessary but the solution is always equal to the original divisor we were trying to calculate because of thm 31.9.
Case 2: The algorithm makes the call Euclid(x, 0)
In this case we know how to provide a solution.
We know the greatest divisor of x is x and the and number divides zero.
Thus the solution to $gcd(x, 0) = 0$ in every case.
The algorithm always makes calls down to the second case.
Example:
$gcd(a, b) = gcd(b, amodb) = ... = gcd(x, 0)$

5.4.2 Extended Euclid's Algorithm Proof

Case 1: If the algorithm gets to the if statement on line 1 then the algorithm returns $(a, 1, 0)$. This is a correct solution since we know from the Euclid's algorithm proof that a will be the gcd of a and b. It also follows the linear combination is correct since $a = a(1) + b(0)$.
Case 2: The algorithm enters line 4 and calculates d' s.t. $d' = gcd(b, amodb)$ so $d' = bx' + (amobd)y'$ on line 5.
Then we have $d = gcd(a, b) = d' = gcd(b, amodb)$.
We want $d = ax + by$, which we can get by rewriting the equation above using thm 3.8.
$d = bx' + a(a - b\lfloor a/b \rfloor)y' = ay' + b(x' - \lfloor a/b \rfloor * y')$
Thus if we choose $x = y'$ and $y = x' - \lfloor a/b \rfloor * y'$ as we do on line 5, we know we will get $d = ax + by$

5.5 Run time

The run time of Euclid's algorithm and Extended Euclid's algorithm is dependent on the number of recursive calls the algorithm makes to itself.
Claim - The algorithm makes at most $log(b)$ recursive calls.
Proof:
Note: the algorithm keeps making recursive calls until it makes one where $b = 0$.
Assume $a \geq b \geq 0$ since if $b \geq a$ the first call will just flip the arguments and if $a = 0$ or $b = 0$ we just get a solution.
Notice that if the above conditions are met the $amodb \leq 1/2 * b$, then the new value used in the recursion for the b parameter is less than for equal to $1/2 * b$.
Then since we make recursive calls until $b = 0$ and each call cuts b to 1/2*b or less the maximum number of recursive calls made by the algorithm is log(b), so the run time of the algorithm is proportional to O(log(b)).

It is worth noting that the run time of the algorithm is also dependent of the size of the integers a and b since it will take longer to calculate $amodb$ for large integers.
The extended algorithm has the same termination conditions so it will run the same number of recursive calls as the original algorithm.

6 Algorithm Discussion

Euclid's algorithm is currently accepted as the fastest way to calculate the greatest common divisor of 2 numbers. The other way we would theoretically try and find them is to find all of the factors of 2 numbers and find the greatest common factor found. This method would be slower than Euclid's since we don't have a fast way to find factors. The algorithm does make one assumption about the input data that causes some limitations. Namely that Euclid's only takes non negative integers as input, this causes no issues with the regular Euclid's

since the greatest common divisor of x is equal to the greatest common divisor of $|x|$ so if it was desired to use Euclid's with negative inputs we can just supply the absolute value and get the correct solution. This also causes an issue that we can solve in the Extended Euclid's, since if a or b is negative we will have an inaccurate linear combination. We can fix this by changing the sign of x if a is less than 0, and changing the sign of y if b is less than 0.
Like many other number theoretic algorithms, Euclid's algorithm wasn't all that useful until long after it was discovered, originally only being very useful for factorization. Many years after its discovery with the invention of computers we found new applications for Euclid's algorithm, the most prevalent of which being used for cryptography. The future of Euclid's algorithm is uncertain, as it hasn't had much recent work since about the 1980's when it was first heavily used for cryptography. Though Euclid's may prove useful in the future of computer science, including future applications in security as the ability to find linear combinations is useful. Some surprising applications of Euclid's include generating musical rhythms and generating linear computer graphics, which shows there may be many abstract uses for the algorithm that are not obvious at first glance.

References

Cormen, Thomas H., et al. Introduction to Algorithms, 3rd Edition. The MIT Press, 2009.

Euclid, and F. L. Thompson. Euclid's Elements. J. Palmer, 1890.

Fournaris, Apostolos P., and O. Koufopavlou. "Applying Systolic Multiplication–Inversion Architectures Based on Modified Extended Euclidean Algorithm for GF(2k) in Elliptic Curve Cryptography." Computers amp; Electrical Engineering, vol. 33, no. 5-6, 5 July 2007, pp. 333–348., doi:10.1016/j.compeleceng.2007.05.001.

Ramachandran, Parthasarathy. "Use of Extended Euclidean Algorithm in Solving a System of Linear Diophantine Equations with Bounded Variables." SpringerLink, Springer, Berlin, Heidelberg, 23 July 2006, link.springer.com/chapter/10.1007/11792086$_1$4.

says:, Liam, et al. "What Is RSA Encryption and How Does It Work?" Comparitech, 24 Oct. 2019, www.comparitech.com/blog/information-security/rsa-encryption/.

Toussaint, Godfried. "The Euclidean Algorithm Generates Traditional Musical Rhythms." BRIDGES: Mathematical Connections in Art, Music and Sci-

ence, 3 Aug. 2005, pp. 47–56.