EYNAS BALKHAIR. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 14, Issue 12, December 2024, pp 32-37

RESEARCH ARTICLE OPEN ACCESS

Euclidean Algorithm Analysis

Eynas Balkhair

Date of Submission: 11-12-2024 Date of acceptance: 23-12-2024

I. II. Introduction and Motivation

Driven by our curiosity to explore more of the mathematical algorithms and the desire and passion to learn more about algorithms and their analysis, we have chosen an extremely important algorithm that finds the greatest common divisor which is of great importance. This algorithm is the Euclidean algorithm that was developed about 300 BC by the Euclid of Alexandra who is an established Greek mathematician (Macardle et al, 2008). The Euclidean algorithm is considered to be among the best algorithms to find the GCD in terms of efficiency up until today. Euclid who developed this algorithm had also developed many relevant and quite efficient algorithms that are widely used by mathematicians and have many applications. However, the greatest common divisor itself holds significant importance. For instance, it is used to simplify fractions as to reduce them to the least possible ratio. Furthermore, it is also used to find co-prime numbers once the GCD is one, as well as to find the greatest possible number of possible intersections in a grid.

The aim of this project is to deepen and further strengthen our understanding of algorithms and their complexity analysis. Furthermore, another objective of this project is to enhance our academic writing skills as well as our logical and algorithmic thinking.

We aim to provide an in-depth coverage of the Euclidean algorithm as to provide a good reference for future researchers and students. By reviewing the literature we get a better understanding of the efficiency of the algorithm and its counterparts. Afterwards, a concrete definition of the problem, the algorithm, analysis and implementation will be presented. We will be using the C++ high-level programming language as to give a clear implementation of the algorithm. We hope this report meets the professional standards and presents the algorithm in the best way possible.

III. Problem Definition

The Greatest Common Divisor

The greatest common divisor (GCD) of two integers that are not zero is the largest natural number that divides both integers (D'angelo & West, 1997). It is also known as the greatest common factor (GCF), highest common factor (HCF), greatest common measure (GCM), and highest common divisor (HCD). While it might be easy to find the greatest divisor of a certain number on its own, the problem gets harder as the number becomes two or sometimes more. That is why there are definitions that do not limit the greatest common divisor to be that of only two numbers but leaves it open-ended as the factor of a group of natural numbers (Struve, Sullivan & Mazzarella, 2008).

IV. Literature Review

Finding the greatest common divisor (GCD) is a problem that has been discussed widely and tackled by many scientists. While the first introduced algorithm was that of a brute force, as time went by, many other algorithms that were developed showed better performance. We will cover three algorithms in this literature review in the following order: Brute Force Algorithm, Binary GCD Algorithm (Stein's Algorithm) and the Euclidean Algorithm. It is important to highlight these algorithms to understand the efficiency and advantage that the Euclidean algorithm has over them.

Firstly, the Brute Force Algorithm suggests a straightforward solution that is not efficient. The idea of this algorithm is start with the largest number as to decrement it until the number that divides both of the numbers is reached (Altarawneh, 2011). Both the time and space complexities of this algorithms are really bad since they grow exponentially as the numbers grow. To further demonstrate the loss let us suppose that two numbers n and m are used as input to this algorithm

and n is found to be larger, then in to give that solution at least n/2 iterations are wasted in this algorithm since a divisor of a number is at most its half. The Brute Force Algorithm is found to perform poorly in terms of time and space complexities.

Secondly, Stein's algorithm shows better results than the Brute Force algorithm. Josef Stein's algorithm that was developed in 1967 is also called the binary GCD algorithm. The algorithm attempts to find the greatest common divisor for two numbers m and n through the following process:

1-If m and n are even then the GCD becomes 2GCD(m/2, n/2);

2-If m is even and n is odd then the GCD becomes GCD(m/2, n);

3-If m and n are odd then the GCD becomes GCD(|m-n|/2, n);

4-GCD(m, 0).

The time complexity of this algorithm is found to be $O(n^2)$ which is an acceptable time complexity of a practical algorithm (Sorenson, 1994). The binary GCD algorithm yields a better time complexity performance.

Thirdly, the Euclidean algorithm has a better time complexity than the Brute Force and the binary GCD algorithms. Euclid who is known as the father of Geometry developed many efficient algorithms in his lifetime that are still in use until today and the GCD Euclidean algorithm is no exception. This widely used algorithm attempts to find the greatest common divisor for two numbers m and n in which m is greater than or equal to n. It finds the GCD through dividing m by n then using the remainder to divide quotient and when a quotient of zero is reached, the number that was divided is the greatest common divisor. Not only does this algorithm eliminates many iterations but it also has a better time complexity of O(log(n)) assuming that n is less than m which are being investigated for the GCD (Ravi, 2007). Euclid's algorithm has a good time complexity.

In conclusion, the three algorithms suggest different ways to find the greatest common factor but show different efficiencies. While the Brute Force algorithm might work fine for small numbers, as the numbers increase, the needed time will greatly increase. On the other hand, Stein's algorithm works better for large numbers and an even better algorithm is Euclid's algorithm. That is why we will further discuss the Euclidean algorithm in this report as follows: The algorithm, its proof, its time complexity and an implementation of it using the C++ programming language.

V. Euclidean Algorithm

The Euclidean algorithm finds the greatest common factor through recursively implementing the following steps on two nonnegative integers m and n that are nonzero, where m>=n:

Euclidean Algorithm

Input: m and n ∈ N
Output: g ∈ N, GCD of m and n
If n=0, g=m
Else → While (n<>0)
{
Remainder = m mod n
m=n
n=Remainder
}
g=m

Simply explained, this algorithm divides the larger number by the smaller number then takes the remainder and divide the smaller number by it until the zero is reached, once the zero is reached then the smaller number that was divided is the greatest common divisor (D'angelo & West, 1997).

Example: GCD (198, 50)
198/50 → 198=50*3+48
50/48 → 50=1*48+2
48/2 → 48=24*2+0
The GCD is then 2 and it divides both 198 and 50

4.1. Proof

The way to go about proving this algorithm is by mathematical induction that is used to prove a theory over natural numbers.

Base case: m and n where m>n , suppose n=0 then we can write m=m*1+0 which shows the GCD is terms of n and m as a integer combination.

Induction step: Assuming that the Euclidean algorithm generates the GCD of m and n where m>=n>=1. Now the Euclidean algorithm gives the first pair as GCD(m, n) which later on becomes GCD (n, q) as the first step can be written like m=an+q for a is a number that belongs to the set of natural numbers. The algorithm can be written as q=m-an and by using the proven proposition of GCD(m, n) = GCD(m-an, n) it is proves that GCD(m, n) = GCD(n, q). Since n>=q which in this step could be less than or equal to zero, we can safely say that after a number of iterations, the

algorithm is bound to produce a pair GCD(p, 0) that holds for p=GCD(m, n)=GCD(n, q) which proves the validity of the algorithm (D'angelo & West, 1997).

4.2. Time complexity analysis

To be as precise as possible, the two notations of the lower bound (best case) Ω(f(n)) and upper bound (worst case) O(f(n)) will be used to give an analysis that is as concise as possible.

The best case is pretty trivial to analyze as it obviously happens when n=m which will yield a time complexity of Ω(1).

The worst case is also easy to realize as it happens when one of the numbers is 1 and the other is greater, in that case the time complexity becomes n(m) when m is the number that is greater than 1. However, this problem can be solved really easily by making an exception before applying the algorithm to immediately announce 1 as the greatest common factor. In this case, the previously stated worst case is eliminated and we are now faced with a new worst case scenario.

The time complexity after adding the modification to the algorithm can be logically arrived to assuming that m is greater than n. Using the lemma that states that for any two integers m and n where m>n then m mod n< m/2, and referring to the pairs that were used in the proof: GCD(m, n) → (n, q) → (q, p). We can see that p<n/2 by the lemma which makes for n being reduced by one half each time. Therefore, the time complexity becomes that of $O(\log_2 n)$ which is also widely written as O(lgn) which is a good indicator of the algorithm's efficiency (Ravi, 1997). This time complexity shows that the algorithm is indeed efficient and feasible.

4.3.

4.4. Implementation

We will present the code first followed with the flowchart and four instances of the output.

A. C++ source code

Euclidean Algorithm C++ implementation

```
#include <iostream>
using namespace std;
int GCD(int m, int n){
        int remainder=0;
        while (n!=0){
                remainder= m%n;
                m=n;
                n=remainder;
        }
        return m;
```

```
}
int main()
{
	int gcd, num1, num2, temp;
	cout<<"GCD"<<endl;
	cout<<"Enter the first number : \n";
	cin>>num1;
	cout<<"Enter the second number: \n";
	cin>>num2;
	if (num1<num2){ //To ensure that the greater number is num1
		temp=num2;
		num2=num1;
		num1=temp;
	}
	if (num1==num2)
		gcd=num1;
		else if (num1==0|| num1==1)
			gcd=num1;
				else gcd=GCD(num1, num2);
cout<<"The greatest commond divisor GCD("<<num1<<", "<<num2<<") is :\n "<<gcd<<endl;
cin>>num2;
}
```

B.

C. Flowchart

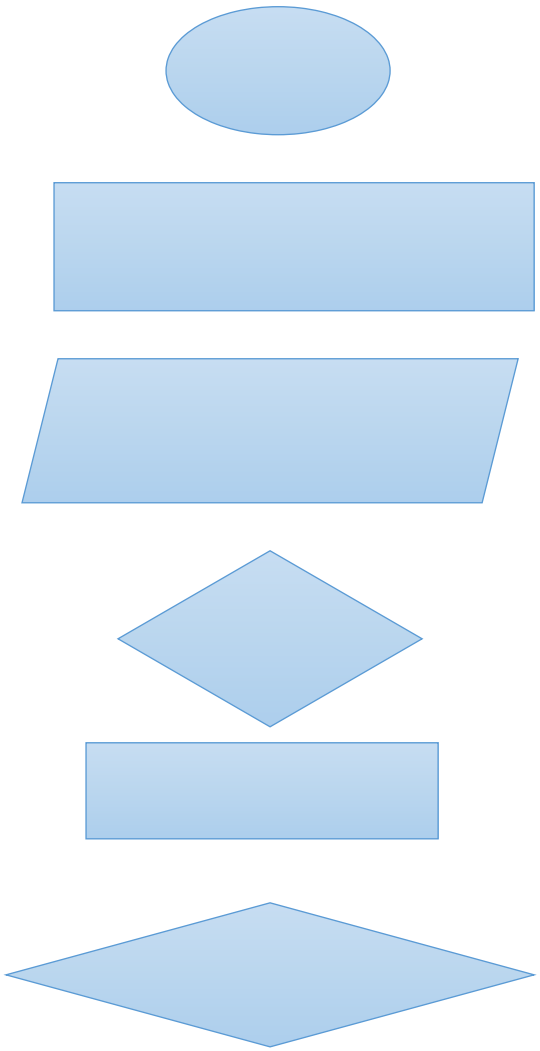

Figure 1: Flowchart for implemented Euclidean algorithm

D. Output instances

```
options | compilation | execution
GCD
Enter the first number :
154
Enter the second number:
35
The greatest commond divisor GCD(154, 35) is :
 7
```

Figure 2: Implementation instance using arbitrary numbers

```
options | compilation | execution
GCD
Enter the first number :
0
Enter the second number:
1
The greatest commond divisor GCD(0, 1) is :
 1
```

Figure 3: Implementation instance of a special case of one and zero

```
options | compilation | execution
GCD
Enter the first number :
7
Enter the second number:
2
The greatest commond divisor GCD(7, 2) is :
 1
```

Figure 4: Implementation instance of two prime numbers that have no GCD

```
options | compilation | execution
GCD
Enter the first number :
1
Enter the second number:
11
The greatest commond divisor GCD(1, 11) is :
 11
```

Figure 5: Implementation instance of a number and 1

4.4 The effect of the Euclidean algorithm's solution in the societal context

This algorithm's solution has facilitated many engineering operations through the effective algorithm that yields the greatest common divisor. The most important and popular applications for the greatest common divisor are found in: Mathematics, encryption and graphics (Koblitz, 1987).

In mathematics, the greatest common divisor is used to simplify fractions and reduce them to the least form. Furthermore, it is also used to find co-prime numbers that are the numbers which are found to be only divisible by 1 mutually.

In encryption, the greatest common divisor is used in the Rivest, Shamir, Adleman (RSA) cryptosystem that uses a public key to encrypt and a private key to decrypt. It is used to find the greatest common divisor for n which is defined in the system and to calculate a number e such that the result of the GCD is one.

In graphics, the greatest common divisor is used to find the largest possible numbers of intersects in a grid. That is, given a certain grid with a number of rows and column, the greatest common divisor can provide the largest number that represents the intersections in that grid.

Overall, the Euclid's algorithm that facilitated calculating the greatest common factor has also facilitated the various applications for the greatest common divisor that affect the society. In an indirect way, the greatest common divisor facilitates encryption which preserves the privacy of the people in society. Moreover, since many computer applications and algorithms require finding the greatest common factor, it then subtly facilitates the life of people indirectly.

VI. Conclusion

After defining the greatest common factor and introducing three algorithms of which we chose the one with the best performance, we presented the best algorithm, its proof, its time complexity analysis and its implementation. The most important highlighted elements were the algorithm's proof and its complexity analysis that was improved greatly with a simple modification.

Overall, an in-depth study for the Euclidean algorithm was presented in this report. The algorithm which showed yielded an impressive result in terms of time complexity analysis also showed a straightforward and direct steps to the algorithm. However, the algorithm needed a small modification to drastically change its worst case time bound. The algorithm was proved to work correctly and was proved by mathematical induction.

While the strength of this report lies in its synthesis and clarity that come with a thorough explanation of every section, it also has points of weakness. The weakness of this study is that it does not cover all the algorithms of the greatest common divisor. This means that there could be a more efficient algorithm or a part of another algorithm that could have been used to enhance the solution to the greatest common factor problem.

Finally, a good recommendation for a future study would be to present a thorough explanation Bishop algorithm for finding the greatest common divisor that is proven to have a better time complexity than the Euclidean algorithm but strangely slows down for small numbers (Altarawneh, 2011). It would also be useful to compare the Euclidean algorithm with its extended version that provides along with the GCD two numbers that are multiplied by n and m respectively to give the GCD.

5. Reflection

Through working on this project we have been able to come closer as a team and work together on producing the best report. This has been quite an enriching experience since we learned a lot about divisibility and mathematical proofs as well as the greatest common factor. Since we find mathematics to be interesting, this project was not boring to research, understand and work with.

Nonetheless, we have faced some problems in finding the time to gather and meet during to the short time frame that was assigned. Moreover, some of us were confused about the tasks that were assigned to them but we were able to overcome these problems in the end. We believe that this has been a rich experience that has also improved our academic writing skills as well as our algorithmic skills. Furthermore, we were exposed to many research papers and improved our skills of finding relevant information that were are looking for through an enormous amount of irrelevant information. Overall, this project has been a great opportunity for us to learn from.

Last but not least, this report is a result of Sumyyah Toonsi and Lamia Balbaid working on the implementation of the algorithm and its complexity analysis as well as Eynas Balkhair writing introductory paragraphs. The report sections were written by Sumyyah Toonsi and the slides to the presentation were made by Eynas Balkhair and Lamia Blabaid and revised by Sumyyah Toonsi.

References

[1]. Ravi, S. S. (2007). Euclid's algorithm cont'd [PDF document]. Retrieved Lecture Notes Online Web Site: http://www.albany.edu/~csi503/pdfs/.

[2]. Altarawneh, H. (2011). A comparison of several greatest common divisor (GCD) algorithms. International Journal of Computer Applications, 26(5).

[3]. D'angelo, J. P., & West D. B. (1997). Mathematical thinking: Problem-solving and Proofs. United States of America: Prentice-Hall.

[4]. Koblitz, N. (1987). A course in number theory and cryptography. New York: Springer.

[5]. Macardle, et al. (2008). Scientists: Extraordinary People Who Altered the Course of History. New York: Metro Books.
[6]. Soernson, J. (1994). Two fast GCD algorithms. Journal of Algorithms, 16.
[7]. Struve, K.R. & Sullivan, M. & Mazzarella, J. (2008). Elementary and intermediate Algebra. (n.p.): Pearson Education.