

WEB3CLUBS FOUNDATION LIMITED

Course Instructor: DR. Cyprian Omukhwaya Sakwa
PHONE: +254723584205 Email: cypriansakwa@gmail.com

Foundational Mathematics for Web3 Builders

Implemented in RUST

Lecture 30

July 8, 2024

Euclidean algorithm

- The Euclidean Algorithm, which allows us to compute gcd of numbers without factoring, is a very useful algorithms in number theory. For instance, it is useful in cryptographic situations where the numbers often have several hundreds of digits and are hard to factor.

Euclidean Algorithm is also called Euclid's algorithm.very important.

Definition 5 (Division Algorithm)

Let $a, b \in \mathbb{Z}$ with $b > 0$. Then a divided by b has quotient q and remainder r means that

$$a = b \cdot q + r \quad \text{with} \quad 0 \leq r < b.$$

- It can be shown that both the quotient and the remainder always exist and are unique, as long as the divisor is not 0. Division algorithm is not actually an algorithm, but this is this theorem's traditional name.

Example 4 Division

$$34 = 5 \times \underline{6} + \underline{4}$$

- Suppose we want to find the greatest common divisor of 36 and 60 using division algorithm, divide 36 into 60 and note the quotient q_1 and remainder r_1 . Then divide r_1 into q_1 to obtain the new quotient q_2 and new remainder r_2 . Continue this process until eventually we get a remainder of 0 as shown below.

Step 1: Apply the division algorithm to 60 and 36.

$$\underline{60} = \underline{36}(1) + \underline{24}$$

Step 2: Apply the division algorithm to 36 and 24.

$$\underline{36} = \underline{24}(1) + \underline{12}$$

Step 3: Apply the division algorithm to 24 and 12.

$$\underline{24} = \underline{12}(2) + \underline{0}$$

The $\gcd(60, 36)$ is the last non-zero remainder and it is 12.

To compute the gcd of two integers by Euclidean algorithm is to find the gcd by repeated division with remainder as explained above.

Example 5

Use division algorithm to compute $\gcd(12600, 756)$

Solution

$$12600 = 756(16) + 504$$

$$756 = 504(1) + 252$$

$$504 = 252(2) + 0$$

$$\text{Thus } \gcd(12600, 756) = 252$$

Example 6

Compute $\text{gcd}(758, 121)$

Solution

$$758 = 121(6) + 32$$

$$121 = 32(3) + 25$$

$$32 = 25(1) + 7$$

$$25 = 7(3) + 4$$

$$7 = 4(1) + 3$$

$$4 = 3(1) + 1$$

$$3 = 1(3) + 0$$

$$\text{gcd}(758, 121) = 1$$

Here's the Rust code for implementing the Euclidean Algorithm:

```


1      fn main() {
2          let num1 = 758; // First number
3          let num2 = 121; // Second number
4          let gcd_result = euclidean_algorithm(num1, num2);
5
6          println!("The GCD of {} and {} is {}", num1, num2, gcd_result);
7      }
8
9      fn euclidean_algorithm(mut a: i64, mut b: i64) -> i64 {
10         while b != 0 {
11             let temp = b;
12             b = a % b;
13             a = temp;
14         }
15         a
16     }

```

When run it prints the gcd.

Understanding the Rust code

```
fn euclidean_algorithm(mut a: i64, mut b: i64) -> i64 {  
    while b != 0 {  
        let temp = b;  
        b = a % b;  
        a = temp;  
    }  
    a  
}
```



- `fn euclidean_algorithm(mut a: i64, mut b: i64)` \Rightarrow `i64` defines a function that takes two mutable `i64` integers (`a` and `b`) and returns an `i64` integer.
- while loop runs as long as `b` is not zero.
- In each iteration of the loop'
 - ✓ The value of `b` is stored in a temporary variable `temp`.
 - ✓ `b` is updated to the remainder of `a` divided by `b` (i.e., `a % b`).
 - ✓ `a` is updated to the value of `temp`.

Understanding the Rust code (cont...)

- When b becomes zero, the loop exits, and the function returns the value of a , which is the gcd of the original inputs.

Example Execution

For $\text{num1} = 756$ and $\text{num2} = 12600$:

Initially, $a = 12600$ and $b = 756$

First iteration:

✓ $\text{temp} = 756$ ✓

✓ $b = 12600 \% 756 = 504$

$\%756=504$

✓ $a = 756$ ✓

Third iteration:

✓ $\text{temp} = 252$ ✓

✓ $b = 504 \% 252 = 0$

✓ $a = 252$ ✓

The loop exits and the function returns 252, as the gcd of 12600 and 756.

Second iteration:

✓ $\text{temp} = 504$ ✓

✓ $b = 756$

$\%504=252$

✓ $a = 504$ ✓

Example 7

Find the gcd and lcm (least common multiple) of 98 and 270. Compare the $\text{lcm}(98, 270)$ to the $\text{gcd}(98, 270)$ and the product of 98 and 270.

Solution

$$\text{lcm} = \frac{|n \cdot m|}{\text{gcd}(n, m)} \checkmark$$

$$\underline{98} = 2 \times 7^2 \checkmark$$

$$\underline{270} = 2 \times 3^3 \times 5 \checkmark$$

$$\text{gcd}(98, 270) = 2 \checkmark$$

$$\text{lcm}(98, 270) = 2 \times 3^3 \times 5 \times 7^2 = \underline{13230}$$

$$\text{Clearly, } \text{lcm}(98, 270) = \frac{|98 \times 270|}{\text{gcd}(98, 270)} = \frac{26460}{2} = 13230 \checkmark$$

Generally, given two numbers a and b , we have $\text{lcm}(a, b) = \frac{|a \cdot b|}{\text{gcd}(a, b)}$.

The Rust program first computes the gcd using the Euclidean Algorithm, which is then used to compute the lcm, as shown below:

```

1 fn main() {
2     let num1 = 98;
3     let num2 = 270;
4
5     let lcm_result = lcm(num1, num2);
6
7     println!("The LCM of {} and {} is {}", num1, num2, lcm_result);
8 }
9
10 fn gcd(mut a: u64, mut b: u64) -> u64 {
11     while b != 0 {
12         let temp = b;
13         b = a % b;
14         a = temp;
15     }
16     a
17 }
18
19 fn lcm(a: u64, b: u64) -> u64 {
20     (a * b) / gcd(a, b)
21 }

```

Exercise

1. Use the Euclidean algorithm to compute each of the following gcd's.

a) $\gcd(4115, 22630)$

b) $\gcd(217284, 39504)$

2. A number L is called a common multiple of m and n if both m and n divide L . The smallest such L is called the least common multiple of m and n and is denoted by $\text{LCM}(m, n)$. For example, $\text{LCM}(3, 5) = 15$ and $\text{LCM}(24, 36) = 72$.

- a) Find the following least common multiples:

i) $\text{LCM}(12, 30)$

ii) $\text{LCM}(10, 20)$

iii) $\text{LCM}(60, 180)$

- b) For each of the LCMs that you computed in (a), compare the value of $\text{LCM}(m, n)$ to the values of m, n , and $\gcd(m, n)$. Try to find a relationship.
- c) Use your result in (b) to compute $\text{LCM}(816, 936)$.

Exercise (conti...)

3. Use Rust program to find the LCM of the following

i) $\text{LCM}(12478, 3008)$

ii) $\text{LCM}(107890, 20456)$

iii) $\text{LCM}(6078, 1808)$

The Extended Euclidean Algorithm

$$\gcd(a, b) = \underline{x}a + \underline{y}b$$

- In order to perform computations in modular arithmetic (studied in the next chapter), we have to get familiar with the Extended Euclidean Algorithm.
- The Euclidean Algorithm yields a very useful fact that the $\gcd(a, b)$ can be expressed as a linear combination of a and b . That is, there exist integers x and y such that $\gcd(a, b) = ax + by$. For instance, the gcd of 10 and 4 is 2 and the equation $10x + 4y = 2$ has a solution $(x, y) = (1, -2)$ meaning that $10(1) + 4(-2) = 2$.
- Similarly, the gcd of 195 and 42 is 3 and the equation $195x + 42y = 3$ has a solution $(x, y) = (-3, 14)$ and so $195(-3) + 42(14) = 3$.
- The method for obtaining x and y is called the Extended Euclidean Algorithm.

Example 8

Find $\gcd(765, 364)$ and express it in the form $765x + 364y$

Solution

Using Euclidean Algorithm we get

$$765 = 364(2) + 37$$

$$364 = 37(9) + 31$$

$$37 = 31(1) + 6$$

$$31 = 6(5) + 1$$

$$6 = 1(6) + 0$$

$$\text{Thus } \gcd(765, 364) = 1$$

$$364 - 37(9)$$

$$1 = 31 - 6(5)$$

$$= 31 - [37 - 31(1)](5)$$

Solution (conti...)

Since 1 is the gcd, we apply the extended Euclid's algorithm by first making 1 in second last equation the subject of the formula. Then substituting all the remainders one at a time from bottom going up.

$$1 = 31 - 6(5)$$

$$= 31 - [37 - 31(1)](5) = 31 - 37(5) + 31(5) = -37(5) + 31(6)$$

$$= -37(5) + [364 - 37(9)](6) = -37(5) + 364(6) - 37(54)$$

$$= 364(6) - 37(59) = 364(6) - [765 - 364(2)](59)$$

$$= 364(6) - 765(59) + 364(118) = -765(59) + 364(124)$$

$$\text{Thus } 1 = 765(-59) + 364(124).$$

Here's the Rust code that implements the Extended Euclidean Algorithm:


```

1 fn main() {
2     let a = 60; // First number
3     let b = 33; // Second number
4
5     let (gcd, x, y) = extended_euclidean(a, b);
6
7     println!("The GCD of {} and {} is {}", a, b, gcd);
8     println!("The coefficients x and y are {} and {} respectively", x, y);
9 }
10
11 fn extended_euclidean(a: i64, b: i64) -> (i64, i64, i64) {
12     if b == 0 {
13         return (a, 1, 0);
14     }
15
16     let (gcd, x1, y1) = extended_euclidean(b, a % b);
17
18     let x = y1;
19     let y = x1 - (a / b) * y1;
20
21     (gcd, x, y)
22 }

```

$$\begin{aligned}
 (8, 0) \quad \text{gcd}(8, 0) &= 8 \\
 x &= 1, y = 0 \\
 8 &= 8(1) + 0(0)
 \end{aligned}$$

$$(8, 0) \quad \text{gcd}(8, 0) = 8(1) + 0(y)$$

Understanding the Rust code

```
11 fn extended_euclidean(a: i64, b: i64) -> (i64, i64, i64) {  
12     if b == 0 {  
13         return (a, 1, 0);  
14     }  
15  
16     let (gcd, x1, y1) = extended_euclidean(b, a % b);  
17  
18     let x = y1;  
19     let y = x1 - (a / b) * y1;  
20  
21     (gcd, x, y)  
22 }
```

- `fn extended_euclidean(a: i64, b: i64) -> (i64, i64, i64)` defines a function that takes two i64 integers (a and b) and returns a tuple of three i64 integers.
- Base case: If b is zero, the function returns (a, 1, 0) because:
 - ✓ The GCD of a and 0 is a
 - ✓ The coefficients are 1 and 0 because $a \cdot 1 + 0 \cdot 0 = a$

Exercise 2

1. Find a solution in integers to the following equations

a) $18107x + 3292y = \gcd(18107, 3292)$

b) $24690x + 135780y = \gcd(24690, 135780)$

2. Use Rust program to compute $g = \gcd(a, b)$ and integer solutions to $ax + by = g$ for the following pairs (a, b) .

a) $(9954, 810)$

b) $(5835, 2505)$

c) $(1452, 550)$

d) $(22241739, 19848039)$

e) $(63750, 16774)$

f) $(2827, 3364)$