EDITORIAL

# Ways of Euclidean Algorithmic Thinking

Alexander Ramm

**Alexander Ramm. Ways of Euclidean Algorithmic Thinking. J Pure AppliedMathematics. 2021; 5(1):1-1.**

Initially the Euclidean or Euclid's Algorithm is named after the ancient Greek mathematician Euclid, who first described it in his Elements (c. 300 BC).

**Euclid Algorithm** is based on the below facts.

If we subtract a smaller number from a larger (we reduce a larger number), GCD doesn't change. So if we keep subtracting repeatedly the larger of two, we end up with GCD.

Now instead of subtraction, if we divide the smaller number, the algorithm stops when we find remainder 0.

## Denoted Euclid Domain:

The fundamental theorem of arithmetic applies to any Euclidean domain: Any number from a Euclidean domain can be factored uniquely into irreducible elements. Any Euclidean domain is a unique factorization domain (UFD), although the converse is not true. The Euclidean domains and the UFD's are subclasses of the GCD domains, domains in which a greatest common divisor of two numbers always exists .In other words, a greatest common divisor may exist (for all pairs of elements in a domain), although it may not be possible to find it using a Euclidean algorithm. A Euclidean domain is always a principal ideal domain (PID), an integral domain in which every ideal is a principal ideal. Again, the converse is not true: not every PID is a Euclidean domain.

## Formal Way: Euclidean algorithm

Input Two positive integers, a and b.
Output the greatest common divisor, g, of a and b.
Internal computation
If a<b, exchange a and b.
Divide a by b and get the remainder, r. If r=0, report b as the GCD of a and b.
Replace a by b and replace b by r. Return to the previous step.

The Binary GCD algorithm is an optimization to the normal Euclidean algorithm.
The slow part of the normal algorithm are the modulo operations. Modulo operations, although we see them as O(1)
, are a lot slower than simpler operations like addition, subtraction or bitwise operations. So it would be better to avoid those.
It turns out, that you can design a fast GCD algorithm that avoids modulo operations. It's based on a few properties:

• If both numbers are even, then we can factor out a two of both and compute the GCD of the remaining numbers: gcd(2a,2b)=2gcd(a,b)

• If one of the numbers is even and the other one is odd, then we can remove the factor 2 from the even one: gcd(2a,b)=gcd(a,b) if b

• If both numbers are odd, then subtracting one number of the other one will not change the GCD: gcd(a,b)=gcd(b,a −b) (this can be proven in the same way as the correctness proof of the normal algorithm)

## Extended way: Euclidean Algorithm

Extended Euclidean algorithm also finds integer coefficients x and y such that:
ax + by = gcd(a, b)
Examples:
Input: a = 30, b = 20
Output: gcd = 10
x = 1, y = -1
(Note that 30*1 + 20*(-1) = 10)
Input: a = 35, b = 15
Output: gcd = 5
x = 1, y = -2
(Note that 35*1 + 15*(-2) = 5)
The extended Euclidean algorithm updates results of gcd(a, b) using the results calculated by recursive call gcd (b%a, a).

*Correspondence: Alexander Ramm, Professor, Department of Mathematics, Kansas State University, USA,. E-mail ramm@ksu.edu*
*Received: January 25, 2021, Accepted: January 27, 2021, Published: January 29, 2021*