Euclid's Algorithm for GCD

Nitin Verma
mathsanew.com

March 23, 2021

The *Euclid's Algorithm* for computing the gcd of two non-negative integers utilizes the following theorem.

Lemma 1. *Say a and b are two non-negative integers with $a \geq b$. Then, the set of* Common-Divisors *of (a, b) is exactly same as the set of* Common-Divisors *of $(a \bmod b, b)$.*

Proof. Say, d is a Common-Divisor of a and b. That is, $d \mid a$ and $d \mid b$. Due to the *Division Theorem*, there exist unique integers q and r (denoted $a \bmod b$), $0 \leq r < b$, such that: $a = qb + r = qb + a \bmod b$.

Thus, $a \bmod b = a - qb$. Since, $d \mid a$ and $d \mid b$, we must have, $d \mid (a - qb)$, i.e. $d \mid (a \bmod b)$.

So, every common-divisor of a and b must also divide $a \bmod b$, and so it is also a common-divisor of $a \bmod b$ and b.

Similarly, if d' is a common-divisor of $a \bmod b$ and b, d' must also divide $qb + a \bmod b$, which equals a. So, every common-divisor of $a \bmod b$ and b is also a common-divisor of a and b. □

Due to above, the greatest integer in the set of common-divisors, the GCD, for the two pairs must also be same. Thus:

Theorem 2. *Say a and b are two non-negative integers with $a \geq b$. Then,*

$$gcd(a, b) = gcd(a \bmod b, b).$$

Note that $0 \leq a \bmod b < b$. Also, $max(a, b) = a$, but $max(a \bmod b, b) = b \leq a$. To find the gcd of (a, b), we can try finding the gcd of smaller integer pair $(a \bmod b, b)$. So, the above theorem helped us in reducing the size of the integers for which gcd is to be found.

The new pair, $(a \bmod b, b)$, still consists of non-negative integers, with $b > a \bmod b$. So, we can again apply above theorem on this new pair and further reduce the size of the integers which we have to deal with. This reduction process can be repeated (until we reach a trivial case), and is what forms the basis of Euclid's Algorithm.

Following is an implementation of this algorithm in C. The inputs A and B are any non-negative integers, and the return value is their gcd.

```
int euclid(const int A, const int B)
{
  int a, b;

  a = A; b = B;              /* (1) */

  while(a != 0 && b != 0)
  {
    if(a > b)
      a = a%b;               /* (2) */
    else
      b = b%a;               /* (3) */
  }

  if(a != 0)
    return a;
  else
    return b;
}
```

Correctness Proof

We will prove the correctness of Euclid's Algorithm by arguing about the above program. We first claim that, the invariants of its loop are:

$P : (a \geq 0 \ \wedge \ b \geq 0)$

$Q : (gcd(a, b) = G),$ where $G = gcd(A, B)$ is a constant.

Before the loop starts, both P and Q hold due to the assignment at point (1).

The loop's condition is:

$$C : (a \neq 0 \ \wedge \ b \neq 0)$$

So, whenever an iteration is entered (condition C true) with P being true, we must have:

$$(a > 0 \ \wedge \ b > 0)$$

The assignment (2) (or (3)) in the iteration will update a (or b) to $a \bmod b$ (or $b \bmod a$), leaving it non-negative. Thus, P will continue to hold after the iteration, and so is a loop-invariant.

Additionally, the change made to a or b by an iteration does not change $gcd(a, b)$ due to theorem 2. Thus, every iteration, though modifies a or b, keeps $gcd(a, b)$ intact. Which means, after any number of iterations, $gcd(a, b)$ would retain its initial value, which is $gcd(A, B) = G$. Thus, after each iteration Q must hold, making it a loop-invariant.

Now we will prove that the loop must terminate. Consider the function $t(a, b) = a + b$. Before the loop starts, $t(a, b) = A + B$. Since the loop maintains invariant P, after each iteration $t(a, b) \geq 0$. Every iteration also strictly decreases a or b, and hence $t(a, b)$. So, $t(a, b)$ attains only non-negative integer values, starting at $A + B$ and strictly decreasing after each iteration. As there are only finite integers between 0 and $A+B$, the iterations cannot go on indefinitely and must terminate.

What will have been achieved when the loop terminates? At that point, condition C must not hold, but P and Q must hold (as they do after each iteration). So, we must have:

$$\begin{aligned} & \neg C \ \wedge \ P \ \wedge \ Q \\ \Leftrightarrow \quad & (a = 0 \ \vee \ b = 0) \ \wedge \ (a \geq 0 \ \wedge \ b \geq 0) \ \wedge \ (gcd(a, b) = G) \end{aligned}$$

Since at least one of a and b is 0, we can assume without loss of generality, $b = 0$. Then,

$$gcd(a, b) = G \quad \Leftrightarrow \quad gcd(a, 0) = G \quad \Leftrightarrow \quad a = G$$

Thus, a must contain the value of $gcd(A, B)$. This completes the proof.

Observations

We now make some more observations. Note that, if $a > b$, the assignment at (2) modifies a to a value less than b, resulting in $b > a$. Similarly, if

$b > a$, assignment at (3) reverses the order to give $a > b$. So, if we have $A \neq B$, i.e. if $a > b$ or $b > a$ when the loop starts, every iteration will do (2) or (3) alternatively, every time reversing their order. Hence, after the last iteration, we must have $a > b$ (or $b > a$), with b (or a) being 0, and a (or b) being the gcd G.

The case of $A = B > 0$ would just result in a single iteration making $b = 0$ and leaving a at $A = G$.

For an example run of this program, consider $A = 24, B = 81$. The sequence of values taken by a and b will be:

a	24	24	6	6	0
b	81	9	9	3	3

resulting in b containing the gcd 3 at the end.

A Generic Loop-Invariant

Suppose, there is a predicate $R(x)$ over non-negative integers x with the property that, for any x and y with $x \geq y > 0$, whenever $R(x) \wedge R(y)$ holds, then $R(x \bmod y)$ must also hold. This can be denoted as:

$$(x \geq y > 0) \ \wedge \ R(x) \ \wedge \ R(y) \ \Rightarrow \ R(x \bmod y) \tag{1}$$

Say, one such predicate R holds true for variables a and b ($R(a) \wedge R(b)$) in Euclid's Algorithm just before the loop starts, i.e. when $a = A, b = B$. Any iteration of this loop simply modifies a (or b) to $a \bmod b$ (or $b \bmod a$), thus maintaining $R(a) \wedge R(b)$ due to property (1). Hence, we have found a generic loop-invariant, $R(a) \wedge R(b)$, in the Euclid's Algorithm.

Now, upon loop termination, when $a = gcd(A, B)$ and $b = 0$ (the other case is similar), $R(a) \wedge R(b)$ must still hold, implying $R(gcd(A, B)) \wedge R(0)$. Thus, Euclid's Algorithm has helped us prove the following theorem.

Theorem 3. *Suppose $R(x)$ is a predicate over non-negative integers x with the property* (1)*. Then, for any non-negative integers A and B:*

$$R(A) \wedge R(B) \ \Rightarrow \ R(gcd(A, B))$$

Now, say there is a predicate $R'(x)$ over non-negative integers x with a slightly different property:

$$(x \geq y > 0) \ \wedge \ R'(x) \ \wedge \ R'(y) \ \Rightarrow \ R'(x - y) \tag{2}$$

Then, repeated application of this property shows that for any integer $i \geq 1$ such that $x \geq iy$, $R'(x - iy)$ must also hold. Also, $x = qy + (x \bmod y)$ for some integer q (Division Theorem) and here $q \geq 1$, $x \geq qy$. This implies that $R'(x - qy) = R'(x \bmod y)$ must also hold. In other words, R' also satisfies property (1), and hence can benefit from theorem 3.

Corollary 4. *Suppose $R'(x)$ is a predicate over non-negative integers x with the property* (2)*. Then, for any non-negative integers A and B:*

$$R'(A) \wedge R'(B) \;\Rightarrow\; R'(gcd(A, B))$$

We can use the above theorem to help us prove other relations which deal with integers and their gcd. Two such examples are presented below.

Bézout's Identity

For any integers i and j, we say k is a *Linear Combination* of i and j if there exist integers x and y such that $k = ix + jy$. Say, we are given two non-negative integers A and B. Consider two integers a and b with $a \geq b > 0$, each of which is a linear combination of A and B. It is easy to prove that $a \bmod b$, which is actually $a - qb$ for some integer q (Division Theorem), must also be a linear combination of A and B.

For any non-negative integer z, let us define a predicate R as:

$$R(z) : (z \text{ is a linear combination of } A \text{ and } B)$$

Due to our observation in the last paragraph, we can say:

$$(a \geq b > 0) \;\wedge\; R(a) \;\wedge\; R(b) \;\Rightarrow\; R(a \bmod b)$$

So, due to theorem 3:

$$R(A) \wedge R(B) \;\Rightarrow\; R(gcd(A, B))$$

Note that $R(A)$ and $R(B)$ hold trivially. So, $R(gcd(A, B))$ must also hold. That is, $gcd(A, B)$ is a linear combination of A and B. This proves the Bézout's Identity:

Theorem 5 (Bézout's Identity)**.** *For any non-negative integers A and B, there exist integers x, y such that:*

$$Ax + By = gcd(A, B)$$

Divisors of Fibonacci Numbers

For integers $i \geq 0$, say F_i denotes the i^{th} *Fibonacci Number* ($F_0 = 0, F_1 = 1, F_2 = 1$ etc). It is a known property about Fibonacci Numbers that, for any non-negative integers a and b with $a \geq b$, and any integer d, if $d \mid F_a$ and $d \mid F_b$, then $d \mid F_{a-b}$. For a given integer d, we can define a predicate over non-negative integers x, $R'(x) : (d \mid F_x)$. This predicate in fact satisfies property (2).

So, applying the corollary 4, we get, for any non-negative integers A and B:

$$R'(A) \land R'(B) \;\Rightarrow\; R'(gcd(A, B))$$

That means, if $d \mid F_A$ and $d \mid F_B$, then $d \mid F_{gcd(A,B)}$, for any integer d.

There is a proof about Fibonacci Numbers ([1]) which did such reasoning based on Euclid's Algorithm.

■

References

[1] Proof Wiki. *GCD of Fibonacci Numbers.*
`https://proofwiki.org/wiki/GCD_of_Fibonacci_Numbers`.