

10

The Euclidean Algorithm

Division

Number theory is the mathematics of integer arithmetic. In this chapter we will restrict ourselves to integers, and in particular we will be concerned primarily with **positive** integers. Every thing that we will do follows from the theorem known as *The Division Algorithm* (note that the theorem itself is not an algorithm).

Theorem: Given two integers n and d , with d positive, there are integers q and r with $0 \leq r < d$ such that $n = d \cdot q + r$.

q is called the *quotient* of n from division by d (the *divisor*) and r is called the *remainder*. $r = 0$ if and only if d divides n evenly. This theorem should seem obvious to anyone competent at grade school arithmetic. If the proof and its discussion is a little too complex, at this point in your development, feel free to go to the next section (Division Rules). The usual proof of this is to use the principle of induction to prove the well ordering of the positive integers; that is that any non-empty set of positive integers has a least element. (We will prove this in section 11.) Then we look at the numbers $n, n - d, n - 2d, \dots$, and apply the well ordering property. Here we are very close to an algorithm for finding q and r . Also, note that this proof requires that n is positive but the proof that we will use will also show that this is not a problem. The proof that we will do here is by induction and does not use the well ordering property.

Proof of The Division Algorithm:

Given integers n and d ($d > 0$) we want to show that there are unique integers q and r , with $0 \leq r < d$ such that $n = q \cdot d + r$. If $n = 0$ then $q = 0$ and $r = 0$. Suppose the theorem holds for positive n . Then given $n < 0$ and $d > 0$, there exists q and r , with $0 \leq r < d$, such that $-n = q \cdot d + r$. Hence $n = -q \cdot d - r$. If $r = 0$ then we are done. Otherwise we add and subtract d from the right hand side (which means we have added zero and

maintained the equality) giving us $n = -q \cdot d - r + d - d$. Rearranging we get $n = -(q + 1)d + (d - r)$. This is the form we are looking for with remainder $(d - r)$ and with $0 < (d - r) < d$ (we have already done the case where the remainder is zero). Hence from now on we can (and will) assume $n > 0$. Next we will show that if q and r exists, they must be unique. Suppose otherwise that $n = q_1 \cdot d + r_1$ and that $n = q_2 \cdot d + r_2$. Subtracting the first equation from the second (thus eliminating n) and rearranging we get $(q_1 - q_2) \cdot d = r_2 - r_1$. This implies d divides $r_2 - r_1$. Since both r_2 and r_1 are positive and each is less than d their difference is between $-d$ and d exclusive. Hence $r_2 - r_1 = 0$ and thus $r_2 = r_1$. This forces $q_1 = q_2$. Thus q and r , if they exists are unique. If $d = 1$ then choose $q = n$ and $r = 0$ and we are done. Now fix d as a constant greater than 1, and we will use induction on the size of n to prove the theorem. Suppose $n = 1$, then since $d > 1$, we $q = 0$ and $r = 1$ satisfy the theorem. Now—this is the induction hypothesis—assume that the theorem holds for $n = k$. (n is already in use here so we use k in n 's usual role in induction.) Set $n = k + 1$. By the induction hypothesis there are a unique q and r such that $k = d \cdot q + r$ with $0 \leq r < d$. Therefore, $k + 1 = d \cdot q + r + 1$. Now if $r + 1$ is less than d , we are done. Otherwise since $r < d$, we must have $r + 1 = d$. Then $k + 1 = d \cdot q + r + 1 = d \cdot q + d$. That is, $k + 1 = (d + 1) \cdot q$. In this last case $r = 0$ and we are done.

Division Rules

Remember, $n|a$ means that n divides a , or more specifically (repeating the definition from Section 3) that there is an integer m such that $a = n \cdot m$. Similarly $n|a, b$ means that n divides both a and b . The following two division properties will be useful to us. We'll refer to these as *the division rules*.

1. If $n|a, b$ then $n|a, b-a$.
2. If $n|a, b-a$ then $n|a, b$.

Again: $X \equiv Y \pmod{n} \Leftrightarrow n \mid X - Y$

□ **Exercise 1** Prove the two division rules given above.

Given two positive integers, m and n , their common divisors are the integers that divide both of them. There can only be a finite collection of such integers and therefore there must be a greatest such integer: *the greatest common divisor*. We can denote the greatest common divisor of m and n by $\text{GCD}(m,n)$. However, we will generally follow the practice typical in number theory, where it is denoted by just (m,n) . The division rules above imply $(n,m) = (m,n-m)$.

Euclid's Algorithm

This last statement is the key to an algorithm. Given a problem: find (n,m) (where $n > m$) we can reduce the problem to $(m,n-m)$. For example, if we want to solve $(95,80)$ we have that it is the same as $(80,15)$ and by the same logic that is the same as $(65,15)$, which becomes $(50,15)$, which in turn is $(35,15) = (20,15) = (15,5) = (10,5) = (5,5) = 5$. Obviously, we have the makings of an algorithm. The main question is what is a termination criteria for the algorithm that will always work. If we pursue the above strategy of always replacing n by $n-m$ we must eventually reach 0 ¹. If we in turn accept $(m,0) = m$ then we are finished. Indeed m does divide 0 in that $m \cdot 0 = 0$. In the above example we would continue $(5,5) = (5,0) = 5$.² We will call the

¹Actually we might reach a negative number. However, we simply rules this out. If the two numbers are unequal we replace the larger by a smaller positive value. This cannot continue forever. Eventually the two values will be equal and we replace one by zero.

²We could always terminate at (x,x) , however going to $(x,0)$ only adds one step, and it is also a good termination criteria for later versions of the algorithm.

algorithm *Euclid's Algorithm* since it is closer to what Euclid had than the modern version *The Euclidean Algorithm* which will follow later.

Euclid's Algorithm

- I. Input positive integers n, m with $n \geq m$.
- II. If $m = 0$ then GCD is n ; we are done.
- III. $n \leftarrow n - m$.
- IV. If $m > n$ then $n \leftrightarrow m$ (exchange m and n).
- V. Goto 2.

As before, it is good programming practice to rewrite the algorithm without the goto statement. That will be done twice, first as an exercise and second, recursively in the next section.

□ **Exercise 2** Rewrite Euclid's algorithm without the goto statement.

It is not difficult to rewrite Euclid's algorithm into a recursive form.

Euclid's Recursive Algorithm

- I. Input positive integers m, n .
 - II. Function GCD(n, m): {This is the recursive procedure.}
 If $1.n < m$ then $m \leftrightarrow n$.
 If $m = 0$ then GCD $\leftarrow n$. Stop. {This is the termination criteria.}
 Call GCD($m, n-m$). {This is the recursive call.}
- End Function.

In this instance GCD(x, y) is a function: it returns a value $z = \text{GCD}(x, y)$. The structure shown above actually works. Note that the statement Call GCD, which calls the function, is

within the function. The algorithm can be coded with exactly the structure given here in many programming languages including Pascal, Java, C, Ada, and Lisp.

□ **Exercise 3** Run through Euclid's Recursive Algorithm for inputs 80, 95; 55, 49; 144, 89.

The Euclidean Algorithm

The Euclidean algorithm is a slight modification of what we have as Euclid's algorithm. It is a simplification in that it usually requires fewer steps to run, but it is a complication in that it replaces subtraction with division. The key insight is that given a problem like (80, 15) instead of proceeding to $(65, 15) = (50, 15) = (35, 15) = (20, 15) = (15, 5)$, we can go from (80, 15) to $(15, 80 \bmod 15)$ by using the division algorithm.

We are viewing the Euclidean algorithm as a speed-up version of Euclid's algorithm. Instead we could proceed by replacing the division rules by the following two rules and go directly to the Euclidean algorithm:

1. $n \mid x, y \Rightarrow n \mid x, (x \bmod y)$
2. $n \mid y, (x \bmod y) \Rightarrow n \mid x, y$

Collectively these two laws imply that $(x, y) = (y, x \bmod y)$. Since $x \bmod y$ is the remainder of x upon division by y , $x \bmod y$ is smaller than y and thus $(y, x \bmod y)$ is a reduction of the (x, y) problem.

The Euclidean Algorithm is stated similarly to Euclid's algorithm:

The Euclidean Algorithm

- I. Input positive integers n, m with $n \geq m$.
- II. If $m = 0$ then GCD is n ; we are done.
- III. $n \leftarrow n \bmod m$.
- IV. $n \leftrightarrow m$.
- V. Goto 2.

The difference between this algorithm and Euclid's algorithm is that in statement three we have $n \leftarrow n \bmod m$ whereas before we had $n \leftarrow n - m$. In this case when we get to statement four, we do not need to test whether m is greater than n because by definition we know that $n \bmod m$ is less than m . As before we should rewrite the algorithm to get rid of the goto statement. One way to do that is to write the algorithm recursively:

The Recursive Euclidean Algorithm

- I. Input positive integers n, m with $n > m$.
 - II. Function GCD(n, m)
 - If $m = 0$ then GCD = n ; Stop.
 - Call GCD($m, n \bmod m$).
- End Function.

This algorithm not only works but it is elegant and it works efficiently.

- **Exercise 4** Use the recursive Euclidean algorithm to solve (90, 12); (194, 78); (144, 89). Whereas it is true that the Euclidean algorithm can be shown to be

efficient, the same analysis also demonstrates that the worst case is where n and m are consecutive Fibonacci numbers such as 89 and 144.¹

□ **Exercise 5** The least common multiple of two numbers x, y is denoted $\text{LCM}(x,y)$. It is the smallest positive integer evenly divided by both numbers. Prove that
$$\text{LCM}(x,y) = \frac{x \cdot y}{\text{GCD}(x,y)} .$$

□ **Exercise 6** It is not hard to guess the meaning of $\text{GCD}(a,b,\dots,m)$ where there are more than two numbers.² Show that when there are more than two numbers that the following recursive definition is correct: $\text{GCD}(a,b,\dots,m,n) = \text{GCD}(\text{GCD}(a,b,\dots,m),n)$.

¹See a full discussion of this in D. E. Knuth's *The Art of Computer Programming: Vol. 1. Fundamental Algorithms*, 2nd. ed. Addison-Wesley. Reading, MA, 1973.

²In mathematics jargon we would say that a, b, \dots, m are *arguments* of the function GCD . However, to non-math majors it sounds like we are talking about disagreements.

1. We have to show that $n|a, b-a$ is **equivalent** to $n|a, b$. $n|a, b$ implies there are positive integers q_1 and q_2 such that $a = q_1 \cdot n$ and $b = q_2 \cdot n$. Hence, $b - a = q_2 \cdot n - q_1 \cdot n = n(q_2 - q_1)$ which means that $n|b - a$. Similarly, suppose that $n|a, b-a$, then there exists positive integers q_1 and q_2 such that $a = q_1 \cdot n$ and $b-a = q_2 \cdot n$. Then $b = b-a + a = q_1 \cdot n + q_2 \cdot n = n(q_1 + q_2)$ and $n|b$.

2. Input positive integers n, m with $n \geq m$.

```

Do While m ≠ 0
    Begin
        n ← n - m
        If m > n then n ↔ m
    End
GCD ← n
    
```

3. The pairs of integers are exactly the same as for the non-recursive algorithm. The code is different and in a sense the logic is different, but the sequence of events are the same.

$(95,80) \rightarrow (80,15) \rightarrow (65,15) \rightarrow (50,15) \rightarrow (35,15) \rightarrow (20,15) \rightarrow (15,5) \rightarrow (10,5) \rightarrow (5,5) \rightarrow (5,0) \rightarrow 5$

$(55,49) \rightarrow (49,6) \rightarrow (43,6) \rightarrow (37,6) \rightarrow (31,6) \rightarrow (25,6) \rightarrow (19,6) \rightarrow (13,6) \rightarrow (7,6) \rightarrow (6,1) \rightarrow (5,1) \rightarrow (4,1) \rightarrow (3,1) \rightarrow (2,1) \rightarrow (1,1) \rightarrow (1,0) \rightarrow 1$

$(144,89) \rightarrow (89,55) \rightarrow (55,34) \rightarrow (34,21) \rightarrow (21,13) \rightarrow (13,8) \rightarrow (8,5) \rightarrow (5,3) \rightarrow (3,2) \rightarrow (2,1) \rightarrow (1,1) \rightarrow (1,0) \rightarrow 1$

4. The pairs of integers are exactly the same as for the non-recursive algorithm. The code is different and in a sense the logic is different, but the sequence of events are the same.

$(90,12) \rightarrow (12,6) \rightarrow (6,0) \rightarrow 6$

$(194,78) \rightarrow (78,38) \rightarrow (38,2) \rightarrow (2,0) \rightarrow 2$

$(144,89) \rightarrow (89,55) \rightarrow (55,34) \rightarrow (34,21) \rightarrow (21,13) \rightarrow (13,8) \rightarrow (8,5) \rightarrow (5,3) \rightarrow (3,2) \rightarrow (2,1) \rightarrow (1,1) \rightarrow (1,0) \rightarrow 1$ (Note that this sequence is exactly as in the previous exercise and algorithm.)

5. Write $x = zx'$ and $y = zy'$ where z is the $\text{GCD}(x,y)$. In particular, x' and y' can not have any remaining common factors. $x'y'z$ is a common multiple of both x and y since it is equal to xy' and $x'y$. Also, it must be the least common multiple, since any common multiple of x and y must contain x' , y' , and z and these three terms have no common

factors. However, $x'y'z = \frac{x'zy'z}{z} = \frac{xy}{\text{GCD}(x,y)}$.

6. $\text{GCD}(a,b,c,\dots,n)$ must refer to the greatest divisor of all the numbers a,b,c,\dots,m,n . We want to verify the recursive relation $\text{GCD}(a,b,\dots,m,n) = \text{GCD}(\text{GCD}(a,b,\dots,m),n)$. Suppose that x is a divisor of a,b,c,\dots,m,n , then it is a divisor of n and it is a divisor of a,b,\dots,m . Similarly, if x is a divisor of n and of a,b,\dots,m then it is a common divisor of a,b,c,\dots,m,n .

