The Euclidean Algorithm

In order to find the inverse of an element $m$ in $\mathbb{Z}/n\mathbb{Z}$, we need to find an integer $a$ satisfying the equation

$$am + bn = 1. \tag{1}$$

Here $b$ is some other integer whose value we don't care about. These notes are about solving (1). The first question is when a solution exists.

**Theorem 1.** Suppose $m$ and $n$ are positive integers. Then the equation $am + bn = 1$ has a solution if and only if the greatest common divisor of $m$ and $n$ is 1.

**Proof.** Suppose that the greatest common divisor of $m$ and $n$ is $d$. Then the equation $1 = am + bn$ implies that $d$ must also be a divisor of 1; so $d = 1$, as we wished to show.

Conversely, suppose that the greatest common divisor of $m$ and $n$ is 1. Multiplication by $m$ defines a mapping $\mu$ that carries the finite set $\mathbb{Z}/n\mathbb{Z}$ to itself:

$$\mu(x) = x \cdot m.$$

What we want to show is that 1 is in the image of $\mu$. Because the set is finite, it's enough to show that $\mu$ is one-to-one; for then the image of $\mu$ will also have $n$ elements, and so must be all of $\mathbb{Z}/n\mathbb{Z}$. So suppose $\mu(x) = \mu(y)$; that is, that

$$x \cdot m = y \cdot m.$$

By the distributive law for multiplication, this means that

$$(x - y) \cdot m = 0,$$

or equivalently that $(x - y)m$ is divisible by $n$. Since $m$ and $n$ are assumed to have no common divisors but 1, it follows that $x - y$ must be divisible by $n$; that is, that $x = y$, as we wished to show. □

So suppose the greatest common divisor of $m$ and $n$ is 1; how do we actually find the solution to (1) that the theorem says has to exist? (The proof of the theorem doesn't help.) One approach is trial and error. If $a$ is one solution to (1), then all the numbers $a + xn$ are also solutions (with $b$ replaced by $b - xn$). This means that there must be a solution between 0 and $n - 1$. We can simply test each of these values of $a$ to see whether $xm$ leaves a remainder of 1 on division by $n$. This is reasonable for small $n$, but nasty to do by hand even for $n$ around 100. For $n$ much bigger than $10^9$, it is not possible even by computer. Fortunately there is a much faster way: the Euclidean algorithm. This algorithm begins with two positive integers $x_0 > x_1 > 0$. The first step is to divide $x_1$ into $x_0$, obtaining a quotient $q_0$ and a remainder $x_2$. The remainder is a non-negative integer strictly smaller than $x_1$. If it isn't zero, we can repeat the process using $x_1$ and $x_2$ in place of $x_0$ and $x_1$. Recording all our divisions, we get a series of equations

$$
x_0 - q_0x_1 = x_2 \quad (0 < x_2 < x_1)
$$

$$
x_1 - q_1x_2 = x_3 \quad (0 < x_3 < x_2)
$$

$$
\vdots
$$

(2)
This continues (always with \(x_i\) getting strictly smaller) until finally some \(x_{N+1}\) is zero. The last interesting equation is

\[
x_{N-2} - q_{N-2}x_{N-1} = x_N,
\]

and \(x_N\) divides \(x_{N-1}\). The Euclidean algorithm says first of all that \(x_N\) is the greatest common divisor of \(x_0\) and \(x_1\). That’s not very hard to prove, but I’ll skip the argument.

Now we turn to solving (1). We’re therefore fixing positive integers \(m\) and \(n\) that have greatest common divisor 1. We apply the Euclidean algorithm, beginning with \(x_0 = n\) and \(x_1 = m\). Since the greatest common divisor is 1, \(x_N\) must be equal to 1. The last equation therefore writes 1 as a linear combination of \(x_{N-2}\) and \(x_{N-1}\) with integer coefficients. Similarly, the next to last equation writes \(x_{N-1}\) as a combination of \(x_{N-3}\) and \(x_{N-2}\) with integer coefficients. Plugging this in for \(x_{N-1}\) in the last equation, we get 1 as a combination of \(x_{N-3}\) and \(x_{N-2}\). Continuing back up the line, we end up with 1 as a combination of \(x_0\) and \(x_1\). The process is easier to do than to explain; so here’s an example. Suppose we want to find an inverse for 19 in \(\mathbb{Z}/65\mathbb{Z}\). Applying the Euclidean algorithm to 65 and 19 gives

\[
\begin{align*}
65 - 3 \cdot 19 &= 8 \\
19 - 2 \cdot 8 &= 3 \\
8 - 2 \cdot 3 &= 2 \\
3 - 2 &= 1.
\end{align*}
\]

The last equation writes 1 as a combination of 2 and 3. Use the preceding one to replace the 2 by a combination of 8 and 3, getting 1 as a combination of 8 and 3:

\[
1 = 3 - 2 = 3 - (8 - 2 \cdot 3) = -8 + (1 + 2) \cdot 3 = -8 + 3 \cdot 3.
\]

Next, use the second equation above to replace the 3 by a combination of 8 and 19:

\[
1 = -8 + 3 \cdot 3 = -8 + 3 \cdot (19 - 2 \cdot 8) = 3 \cdot 19 + (-1 - 6) \cdot 8 = 3 \cdot 19 - 7 \cdot 8.
\]

Finally, plug in the first equation above:

\[
1 = 3 \cdot 19 - 7 \cdot 8 = 3 \cdot 19 - 7(65 - 3 \cdot 19) = -7 \cdot 65 + (3 + 21) \cdot 19 = -7 \cdot 65 + 24 \cdot 19.
\]

This equation says that \(\overline{24}\) is the inverse of \(\overline{19}\) in \(\mathbb{Z}/65\mathbb{Z}\).

Here are some things to think about. First, just how fast or slow is this algorithm? That is, given positive integers \(x_0 > x_1\), can you estimate the number of steps \(N\) in terms of the size of \(x_0\)? (The trial-and-error method required \(x_0\) steps, so we’re looking for something better than that.)

Second, this algorithm depends only on a nice notion of division with remainder. Another place where there is such a notion is the collection \(k[x]\) of polynomials over a commutative field \(k\). More or less everything above can be repeated with the integers replaced by \(k[x]\), \(n\) replaced by a polynomial \(p\) of degree \(d > 0\), and \(m\) replaced by another polynomial \(q\) of degree strictly smaller than \(d\). The ring \(\mathbb{Z}/n\mathbb{Z}\) is replaced by \(k[x]/(p)\), consisting of equivalence classes of polynomials modulo the
relation $q_1 \sim q_2$ whenever $q_1 - q_2$ is divisible by $p$. Just as division with remainder in $\mathbb{Z}$ identifies $\mathbb{Z}/n\mathbb{Z}$ with $\{0, \ldots, n-1\}$, division with remainder in $k[x]$ identifies

$$k[x]/(p) \simeq \{ \text{polynomials of degree strictly less than } d \}.$$ 

A result like the Theorem above says that $q$ is invertible in $k[x]/(p)$ if and only if the greatest common divisor of $p$ and $q$ is 1; and in that case the Euclidean algorithm computes the inverse.

To see if you've understood this second thing to think about, try an example. Use the field $\mathbb{R}$ of real numbers, and the polynomial $p = x^2 - 1$. Show how to identify $\mathbb{R}[x]/(p)$ with the field $\mathbb{C}$ of complex numbers. The Euclidean algorithm is supposed to tell you how to compute the inverse of any non-zero complex number. Does this computation have anything to do with what you already know about finding the inverse of a complex number?