The Extended Euclidean Algorithm

As we know from grade school, when we divide one integer by another (nonzero) integer we get an integer *quotient* (the "answer") plus a *remainder* (generally a rational number). For instance,

13/5 = 2 ("the quotient") + 3/5 ("the remainder").

We can rephrase this division, totally in terms of integers, without reference to the division operation:

13 = 2(5) + 3.

Note that this expression is obtained from the one above it by multiplying through by the divisor 5.

We refer to this way of writing a division of integers as the **Division Algorithm for Integers**. More formally stated:

If a and b are positive integers, there exist integers unique non-negative integers q and r so that
$a = qb + r$, where $0 \leq r < b$.

q is called the *quotient* and r the *remainder*.

The *greatest common divisor* of integers a and b, denoted by *gcd(a,b)*, is the largest integer that divides (without remainder) both a and b. So, for example:

gcd(15, 5) = 5, gcd(7, 9) = 1, gcd(12, 9) = 3, gcd(81, 57) = 3.

The gcd of two integers can be found by repeated application of the division algorithm, this is known as the ***Euclidean Algorithm***. You repeatedly divide the divisor by the remainder until the remainder is 0. The gcd is the last non-zero remainder in this algorithm. The following example shows the algorithm.

Finding the gcd of 81 and 57 by the Euclidean Algorithm:

81 = 1(57) + 24
57 = 2(24) + 9
24 = 2(9) + 6
9 = 1(6) + **3**
6 = 2(3) + 0.

It is well known that if the gcd(a, b) = r then there exist integers p and s so that:

p(a) + s(b) = r.

By reversing the steps in the Euclidean Algorithm, it is possible to find these integers p and s. We shall do this with the above example:

Starting with the next to last line, we have:

3 = 9 -1(6)

From the line before that, we see that 6 = 24 - 2(9), so:

3 = 9 - 1(24 - 2(9)) = 3(9) - 1(24).

From the line before that, we have 9 = 57 - 2(24), so:

3 = 3(57 - 2(24)) - 1(24) = 3(57) - 7(24).

And, from the line before that 24 = 81 - 1(57), giving us:

3 = 3(57) - 7(81 - 1(57)) = 10(57) -7(81).

So we have found p = -7 and s = 10.

The procedure we have followed above is a bit messy because of all the back substitutions we have to make. It is possible to reduce the amount of computation involved in finding p and s by doing some auxillary computations as we go forward in the Euclidean algorithm (and no back substitutions will be necessary). This is known as the ***extended Euclidean Algorithm***.

Before presenting this extended Euclidean algorithm, we shall look at a special application that is the most common usage of the algorithm. We will give a form of the algorithm which only solves this special case, although the general algorithm is not much more difficult.

Consider the problem of setting up the Hill cryptosystem. We were forced to do arithmetic modulo 26, and sometimes we had to find the inverse of a number mod 26. This turned out to be a difficult task (and not always possible). We observed that a number x had an inverse mod 26 (i.e., a number y so that xy = 1 mod 26) if and only if gcd(x, 26) = 1. There is nothing special about 26 here, so let us consider the general case of finding inverses of numbers modulo n. The inverse of x exists if and only if gcd(x, n) = 1. We now know that if this is true, there exist integers p and s so that

$$px + sn = 1.$$

But this says that $px = 1 + (-s)n$, or in other words, $px \equiv 1 \pmod{n}$. So, p (reduced mod n if need be) is the inverse of x mod n. The extended Euclidean algorithm will give us a method for calculating p efficiently (note that in this application we do not care about the value for s, so we will simply ignore it.)

The Extended Euclidean Algorithm for finding the inverse of a number mod n.

We will number the steps of the Euclidean algorithm starting with step 0. The quotient obtained at step i will be denoted by q_i. As we carry out each step of the Euclidean algorithm, we will also calculate an auxillary number, p_i. For the first two steps, the value of this number is given: $p_0 = 0$ and $p_1 = 1$. For the remainder of the steps, we recursively calculate $p_i = p_{i-2} - p_{i-1}\, q_{i-2} \pmod{n}$. Continue this calculation for one step beyond the last step of the Euclidean algorithm.

The algorithm starts by "dividing" n by x. If the last non-zero remainder occurs at step k, then if this remainder is 1, x has an inverse and it is p_{k+2}. (If the remainder is not 1, then x does not have an inverse.) Here is an example:

Find the inverse of 15 mod 26.

Step 0:	$26 = 1(15) + 11$	$p_0 = 0$
Step 1:	$15 = 1(11) + 4$	$p_1 = 1$
Step 2:	$11 = 2(4) + 3$	$p_2 = 0 - 1(1) \bmod 26 = 25$
Step 3:	$4 = 1(3) + 1$	$p_3 = 1 - 25(1) \bmod 26 = -24 \bmod 26 = 2$
Step 4:	$3 = 3(1) + 0$	$p_4 = 25 - 2(2) \bmod 26 = 21$
		$p_5 = 2 - 21(1) \bmod 26 = -19 \bmod 26 = 7$

Notice that $15(7) = 105 = 1 + 4(26) \equiv 1 \pmod{26}$.