6 Euclid's Algorithm

In this section, we present Euclid's algorithm for the greatest common divisor of two integers. An extended version of this algorithm will furnish the one implication that is missing in Figure 5.

Reduction. An important insight is Euclid's Division Theorem stated in Section 4. We use it to prove a relationship between the greatest common divisors of numbers j and k when we replace k by its remainder modulo j.

LEMMA. Let $j, k, q, r > 0$ with $k = jq + r$. Then $\gcd(j, k) = \gcd(r, j)$.

PROOF. We begin by showing that every common factor of j and k is also a factor of r. Letting $d = \gcd(j, k)$ and writing $j = Jd$ and $k = Kd$, we get

$$r \;=\; k - jq \;=\; (K - Jq)d.$$

We see that r can be written as a multiple of d, so d is indeed a factor of r. Next, we show that every common factor of r and j is also a factor of k. Letting $d = \gcd(r, j)$ and writing $r = Rd$ and $j = Jd$, we get

$$k \;=\; jq + r \;=\; (Jq + R)d.$$

Hence, d is indeed a factor of k. But this implies that d is a common factor of j and k iff it is a common factor of r and j. □

Euclid's gcd algorithm. We use the Lemma to compute the greatest common divisor of positive integers j and k. The algorithm is recursive and reduces the integers until the remainder vanishes. It is convenient to assume that both integers, j and k, are positive and that $j \leq k$.

```
integer GCD(j, k)
  q = k div j; r = k − jq;
  if r = 0 then return j
           else return GCD(r, j)
  endif.
```

If we call the algorithm for $j > k$ then the first recursive call is for k and j, that is, it reverses the order of the two integers and keeps them ordered as assumed from then on. Note also that $r < j$. In words, the first parameter, j, shrinks in each iterations. There are only a finite number of non-negative integers smaller than j which implies that after a finite number of iterations the algorithm halts with $r = 0$. In other words, the algorithm terminates after a finite number of steps, which is something one should always check, in particular for recursive algorithms.

Last implication. We modify the algorithm so it also returns the integers x and y for which $\gcd(j, k) = jx+ky$. This provides the missing implication in Figure 5.

D'. If $\gcd(a, n) = 1$ then the linear equation $ax+ny = 1$ has a solution.

This finally verifies that the gcd is a test for the existence of a multiplicative inverse in modular arithmetic. More specifically, $x \bmod n$ *is* the multiplicative inverse of a in $\mathbb{Z}_n$. Do you see why? We can thus update the relationship between the statements I, II, III, IV listed at the beginning of Section 5; see Figure 6.

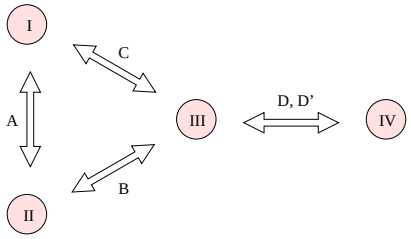

Figure 6: Equivalences between the statements listed at the beginning of Section 5.

Extended gcd algorithm. If $r = 0$ then the above algorithm returns j as the gcd. In the extended algorithm, we also return $x = 1$ and $y = 0$. Now suppose $r > 0$. In this case, we recurse and get

$$\begin{aligned} \gcd(r, j) &= rx' + jy' \\ &= (k - jq)x' + jy' \\ &= j(y' - qx') + kx'. \end{aligned}$$

We thus return $g = \gcd(r, j)$ as well as $x = y' - qx'$ and $y = x'$. As before, we assume $0 < j \leq k$ when we call the algorithm.

```
integer³ xGCD(j, k)
  q = k div j; r = k − jq;
  if r = 0 then return (j, 1, 0)
           else (g, x', y') = xGCD(r, j);
                return (g, y' − qx', x')
  endif.
```

To illustrate the algorithm, we run it for $j = 14$ and $k = 24$. The values of $j, k, q, r, g = \gcd(j,k), x, y$ at the various levels of recursion are given in Table 2.

j	k	q	r	g	x	y
14	24	1	10	2	-5	3
10	14	1	4	2	3	-2
4	10	2	2	2	-2	1
2	4	2	0	2	1	0

Table 2: Running the extended gcd algorithm on $j = 14$ and $k = 24$.

Computing inverses. We have established that the integer a has a multiplicative inverse in $\mathbb{Z}_n$ iff $\gcd(a,n) = 1$. Assuming $n = p$ is a prime number, this is the case whenever $a < p$ is positive.

COROLLARY. If p is prime then every non-zero $a \in \mathbb{Z}_p$ has a multiplicative inverse.

It is straightforward to compute the multiplicative inverse using the extended gcd algorithm. As before, we assume p is a prime number and $0 < a < p$.

```
integer INVERSE(a, p)
  (g, x, y) = xGCD(a, p);
  assert g = 1; return x mod p.
```

The assert statement makes sure that a and p are indeed relative prime, for else the multiplicative inverse would not exist. We have seen that x can be negative so it is necessary to take x modulo p before we report it as the multiplicative inverse.

Multiple moduli. Sometimes, we deal with large integers, larger then the ones that fit into a single computer word (usually 32 or 64 bits). In this situation, we have to find a representation that spreads the integer over several words. For example, we may represent an integer x by its remainders modulo 3 and modulo 5, as shown in Table 3. We see that the first 15 non-negative integers correspond to different pairs of remainders. The generalization of this insight to relative prime numbers m and n is known as the

x	0	1	2	3	4	...	13	14	15
$x \bmod 3$	0	1	2	0	1	...	1	2	0
$x \bmod 5$	0	1	2	3	4	...	3	4	0

Table 3: Mapping the integers from 0 to 15 to pairs of remainders after dividing with 3 and with 5.

CHINESE REMAINDER THEOREM. Let $m, n > 0$ be relative prime. Then for every $a \in \mathbb{Z}_m$ and $b \in \mathbb{Z}_n$, the system of two linear equations

$$\begin{aligned} x \bmod m &= a; \\ x \bmod n &= b \end{aligned}$$

has a unique solution in $\mathbb{Z}_{mn}$.

There is a further generalization to more then two moduli that are pairwise relative prime. The proof of this theorem works as suggested by the example, namely by showing that $f : \mathbb{Z}_{mn} \to \mathbb{Z}_m \times \mathbb{Z}_n$ defined by

$$f(x) = (x \bmod m, x \bmod n)$$

is injective. Since both $\mathbb{Z}_{mn}$ and $\mathbb{Z}_m \times \mathbb{Z}_n$ have size mn, this implies that f is a bijection. Hence, $(a,b) \in \mathbb{Z}_m \times \mathbb{Z}_n$ has a unique preimage, the solution of the two equations.

To use this result, we would take two large integers, x and y, and represent them as pairs, $(x \bmod m, x \bmod n)$ and $(x \bmod m, x \bmod n)$. Arithmetic operations can then be done on the remainders. For example, x times y would be represented by the pair

$$\begin{aligned} xy \bmod m &= [(x \bmod m)(y \bmod m)] \bmod m; \\ xy \bmod n &= [(x \bmod n)(y \bmod n)] \bmod n. \end{aligned}$$

We would choose m and n small enough so that multiplying two remainders can be done using conventional, single-word integer multiplication.

Summary. We discussed Euclid's algorithm for computing the greatest common divisor of two integers, and its extended version which provides the missing implication in Figure 5. We have also learned the Chinese Remainder Theorem which can be used to decompose large integers into digestible junks.