Modular multiplicative inverse

From Wikipedia, the free encyclopedia

In modular arithmetic, the **modular multiplicative inverse** of an integer *a* modulo *m* is an integer *x* such that

$$a\,x \equiv 1 \pmod{m}.$$

That is, it is the multiplicative inverse in the ring of integers modulo *m*, denoted $\mathbb{Z}_m$.

Once defined, *x* may be noted a^{-1}, where the fact that the inversion is m-modular is implicit.

The multiplicative inverse of *a* modulo *m* exists if and only if *a* and *m* are coprime (i.e., if gcd(*a*, *m*) = 1). If the modular multiplicative inverse of *a* modulo *m* exists, the operation of division by *a* modulo *m* can be defined as multiplying by the inverse, which is in essence the same concept as division in the field of reals.

Contents

Example

Suppose we wish to find modular multiplicative inverse *x* of 3 modulo 11.

$$x \equiv 3^{-1} \pmod{11}$$

This is the same as finding *x* such that

$$3x \equiv 1 \pmod{11}$$

Working in $\mathbb{Z}_{11}$ we find one value of *x* that satisfies this congruence is 4 because

$$3(4) = 12 \equiv 1 \pmod{11}$$

and there are no other values of *x* in $\mathbb{Z}_{11}$ that satisfy this congruence. Therefore, the modular multiplicative inverse of 3 modulo 11 is 4.

Once we have found the inverse of 3 in $\mathbb{Z}_{11}$, we can find other values of *x* in $\mathbb{Z}$ that also satisfy the congruence. They may be found by adding multiples of *m = 11* to the found inverse. Generalizing, all possible *x* for this example can be formed from

$$4 + (11 \cdot z), z \in \mathbb{Z}$$

yielding {..., −18, −7, **4**, 15, 26, ...}.

Computation

Extended Euclidean algorithm

The modular multiplicative inverse of *a* modulo *m* can be found with the extended Euclidean algorithm. The algorithm finds solutions to Bézout's identity

$$ax + by = \gcd(a, b)$$

The Wikibook *Algorithm Implementation* has a page on the topic of: ***Extended***

where *a* and *b* are given and *x*, *y* and gcd(*a*, *b*) are the integers that the algorithm discovers. So, since the modular multiplicative inverse is the solution to

Euclidean algorithm

$$ax \equiv 1 \pmod{m},$$

by the definition of congruence, $m \mid ax - 1$, which means that m is a divisor of $ax - 1$. This, in turn, means that

$$ax - 1 = qm.$$

Rearranging produces

$$ax - qm = 1,$$

with *a* and *m* given, *x* the inverse, and *q* an integer multiple that will be discarded. This is the exact form of equation that the extended Euclidean algorithm solves—the only difference being that gcd(*a*, *m*) = 1 is predetermined instead of discovered. Thus, *a* needs to be coprime to the modulus, or the inverse won't exist.

This algorithm runs in time $O(\log(m)^2)$, assuming $|a| < m$, and is generally more efficient than exponentiation.

Using Euler's theorem

As an alternative to the extended Euclidean algorithm, Euler's theorem may be used to compute modular inverse:[1]

According to Euler's theorem, if *a* is coprime to *m*, that is, gcd(*a*, *m*) = 1, then

$$a^{\varphi(m)} \equiv 1 \pmod{m}$$

where φ(*m*) is Euler's totient function. This follows from the fact that *a* belongs to the multiplicative group $(\mathbf{Z}/m\mathbf{Z})^\times$ iff *a* is coprime to *m*. Therefore the modular multiplicative inverse can be found directly:

$$a^{\varphi(m)-1} \equiv a^{-1} \pmod{m}$$

In the special case when m is a prime, the modular inverse is given by the below equation as:

$$a^{-1} \equiv a^{m-2} \pmod{m}$$

This method is generally slower than the extended Euclidean algorithm, but is sometimes used when an implementation for modular exponentiation is already available. Some disadvantages of this method include:

- The value $\varphi(m)$ must be known, whose most efficient computation requires m's factorization. Factorization is widely believed to be a computationally hard problem. However, calculating $\varphi(m)$ is straightforward when the prime factorisation of m is known.
- The relative cost of exponentiation. Though it can be implemented more efficiently using modular exponentiation, when large values of m are involved this is most efficiently computed with the Montgomery reduction method. This algorithm itself requires a modular inverse mod m, which is what was to be calculated in the first place. Without the Montgomery method, we're left with standard binary exponentiation which requires division mod m at every step, a slow operation when m is large. Furthermore, any kind of modular exponentiation is a taxing operation with computational complexity $O(\log \varphi(m)) = O(\log m)$.

Applications

The modular multiplicative inverse has many applications in algorithms, particularly those related to number theory, since many such algorithms rely heavily on the theory of modular arithmetic. As a simple example, consider the *exact division problem* where you have a list of odd word-sized numbers each divisible by k and you wish to divide them all by k. One solution is as follows:

1. Use the extended Euclidean algorithm to compute k^{-1}, the modular multiplicative inverse of k mod 2^w, where w is the number of bits in a word. This inverse will exist since the numbers are odd and the modulus has no odd factors.
2. For each number in the list, multiply it by k^{-1} and take the least significant word of the result.

On many machines, particularly those without hardware support for division, division is a slower operation than multiplication, so this approach can yield a considerable speedup. The first step is relatively slow but only needs to be done once.

See also

- Inversive congruential generator
- Modular arithmetic
- Number theory
- Public-key cryptography
- Rational reconstruction (mathematics)

References

1. ^ Thomas Koshy. Elementary number theory with applications (http://books.google.com/books?id=d5Z5I3gnFh0C&pg=PA346), 2nd edition. ISBN 978-0-12-372487-8. P. 346.

- Weisstein, Eric W., "Modular Inverse" (http://mathworld.wolfram.com/ModularInverse.html), *MathWorld*.

Retrieved from "http://en.wikipedia.org/w/index.php?title=Modular_multiplicative_inverse&oldid=635038970"

Categories: Modular arithmetic
