Reverse ring element modulo

Definition

Suppose we are given a natural module m, and consider the ring formed by the module (ie, consisting of the numbers 0up $m-1$). Then for some elements of this ring can be found **an inverse element** .

The inverse of the number $a$modulo $m$called a number $b$that:

$$a \cdot b \equiv 1 \pmod m,$$

and it is often denoted by a^{-1}.

It is clear that for the zero inverse element does not exist ever, and for the remaining elements of the inverse can exist or not. It is argued that the inverse exists only for those elements $a$that **are relatively prime** to the modulus m.

Consider the following two ways of finding the inverse element employed, provided that it exists.

Finally, consider an algorithm that allows you to find all the numbers back to some modulo linear time.

Finding using the Advanced Euclidean algorithm

Consider the auxiliary equation (relatively unknown $x$and y):

$$a \cdot x + m \cdot y = 1.$$

This linear Diophantine equation of the second order . As shown in the article on the condition $\gcd(a,m)=1$that this equation has a solution which can be found using the Extended Euclidean algorithm (hence the same way, it follows that when $\gcd(a,m) \neq 1$, solutions, and therefore the inverse element does not exist).

On the other hand, if we take from both sides of the residue modulo m, we obtain:

$$a \cdot x = 1 \pmod m.$$

Thus found $x$and will be the inverse of a.

Implementation (including that found $x$necessary to take the modulus $m$and $x$may be negative):

```
int x, y;
int g = gcdex (a, m, x, y);
if (g != 1)
     cout << "no solution";
else {
     x = (x % m + m) % m;
     cout << x;
}
```

Asymptotics of solutions is obtained $O(\log m)$.

Finding the binary exponentiation

We use Euler's theorem:

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

which is true just in the case of relatively prime $a$and m.

Incidentally, in the case of a simple module m, we get an even more simple statement - Fermat's little theorem:

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

Multiply both sides of each equation by a^{-1}, we obtain:

- for any module m:
- $a^{\phi(m)-1} \equiv a^{-1} \pmod m,$
- for a simple module m:
- $a^{m-2} \equiv a^{-1} \pmod m.$

Thus, we got the formula for direct calculation of the inverse. For practical applications typically use efficient algorithm for binary exponentiation , which in this case will bring about for exponentiation $O(\log m)$.

This method appears to be somewhat easier described in the previous paragraph, but it requires knowledge of the values of the Euler function that actually requires factorization module $m$that can sometimes be quite challenging.

If the quotient of the number is known, then this method also works for the asymptotic behavior $O(\log m)$.

Finding all simple modulo a given linear time

Let a simple module m. Required for each number in the interval $[1; m-1]$ to find its inverse.

Applying the above algorithm, we obtain a solution with the asymptotic behavior $O(m \log m)$. Here we present a simple solution to the asymptotic behavior $O(m)$.

Solution is as follows. We denote $r[i]$ the inverse of the required number i modulo m. Then the $i > 1$ following identity holds:

$$r[i] = - \left\lfloor \frac{m}{i} \right\rfloor \cdot r[m \bmod i]. \pmod m$$

Implementation of this amazingly concise solutions:

```
r[1] = 1;
for (int i=2; i<m; ++i)
      r[i] = (m - (m/i) * r[m%i] % m) % m;
```

Proof of this solution is a chain of simple transformations:

We write the value $m \bmod i$:

$$m \bmod i = m - \left\lfloor \frac{m}{i} \right\rfloor \cdot i,$$

whence, taking both sides modulo m, we obtain:

$$m \bmod i = - \left\lfloor \frac{m}{i} \right\rfloor \cdot i. \pmod m$$

Multiplying both sides by the inverse of i the reverse to $(m \bmod i)$ obtain the desired formula:

$$r[i] = - \left\lfloor \frac{m}{i} \right\rfloor \cdot r[m \bmod i], \pmod m$$

QED.