forthright48
learning never ends

Chinese Remainder Theorem Part 2 – Non Coprime Moduli

forthright48 on November 18, 2017

As promised on the last post, today we are going to discuss the "Strong Form" of Chinese Remainder Theorem, i.e, what do we do when the moduli in the congruence equations are not pairwise coprime. The solution is quite similar to the one we have already discussed in the previous post, so hopefully, it will be a lot easier to understand.

Prerequisite

If you haven't read my previous post, you can find it here: Chinese Remainder Theorem Part 1. I strongly suggest you read that one before proceeding onwards.

Problem

Given two sequences of numbers $A = [a_1, a_2, \ldots, a_n]$ and $M = [m_1, m_2, \ldots, m_n]$, find the smallest solution to the following linear congruence equations if it exists:

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\cdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

Note: The elements of M are not pairwise coprime

Working With Two Equations

Before we go on to deal with n equations, let us first see how to deal with two equations. Just like before, we will try to merge two equations into one.

So given the two equation $x \equiv a_1 \pmod{m_1}$ and $x \equiv a_2 \pmod{m_2}$, find the smallest solution to x if it exists.

Existance of Solution

Suppose, $gcd(m_1, m_2) = g$. In that case, the following constraint must be satisfied:
$a_1 \equiv a_2 \pmod{g}$, otherwise there does not exist any solution to x. Why is that?

We know that, $x \equiv a_1 \pmod{m_1}$
or, $x - a_1 \equiv 0 \pmod{m_1}$
or, $m_1 | x - a_1$

Since m_1 divides $x-a_1$, any divisor of m_1 also divides $x-a_1$, including g.

$\therefore x-a_1 \equiv 0 (\mod g)$.

Similarly, we can show that, $x-a_2 \equiv 0 (\mod g)$ is also true.

$\therefore x-a_1 \equiv x-a_2 (\mod g)$
or, $a_1 \equiv a_2 (\mod g)$
or, $a_1-a_2 \equiv (\mod g)$
or $g|(a_1-a_2)$

Therefore, g must divide a_1-a_2, otherwise, there is conflict and no solution exists. From this point, we assume that $g|a_1-a_2$.

Finding Solution

Just like last time, we use Bezout's identity to find a solution to x. From Bezout's identity, we know that there exists a solution to the following equations:

$m_1 p + m_2 q = g$

Since both side is divisible by g, we can rewrite it as:

$$\frac{m_1}{g} p + \frac{m_2}{g} q = 1$$

Using Extended Euclidean Algorithm, we can easily find values of p and q. Simply call the function `ext_gcd(m1/g, m2/g, p, q)` and it should be calculated automatically.

Once we know the value of p and q, we can claim that the solution to x is the following:

$$x = a_1 \frac{m_2}{g} q + a_2 \frac{m_1}{g} p$$

Why is that? Look below.

$$x = a_1 \frac{m_2}{g} q + a_2 \frac{m_1}{g} p$$
$$\text{From Bezout's Identity, we know that } \frac{m_1}{g} p = 1 - \frac{m_2}{g} q$$
$$\text{so, } x = a_1 \frac{m_2}{g} q + a_2 \left(1 - \frac{m_2}{g} q\right)$$

$$\text{or, } x = a_2 + (a_1 - a_2)\frac{m_2}{g}q$$
$$\text{Since, } g|(a_1 - a_2)$$
$$x = a_2 + \frac{a_1 - a_2}{g}m_2 q$$
$$\therefore x \equiv a_2 \pmod{m_2}$$
$$\text{Similarly, } x \equiv a_1 \pmod{m_1}$$

Hence, $x = a_1 \frac{m_1}{g}q + a_2 \frac{m_2}{g}p$ is a valid solution. It may not be the smallest solution though.

Finding Smallest Solution

Let x_1 and x_2 be adjacent two solutions. Then the following are true:

$x_1 \equiv a_1 \pmod{m_1}$
$x_2 \equiv a_1 \pmod{m_1}$
$x_1 \equiv x_2 \pmod{m_1}$
$x_1 - x_2 \equiv 0 \pmod{m_1}$
$m_1 | x_1 - x_2$

Similarly, $m_2 | x_1 - x_2$

Since both m_1 and m_2 divides $x_1 - x_2$, we can say that $LCM(m_1, m_2)$ also divides $x_1 - x_2$.

Since any two adjacent solution of x is divisible by $L = LCM(m_1, m_2)$, then there cannot be more than one solution that lies on the range 0 to $L - 1$. Hence, the following is the smallest solution to x :

$$x \equiv a_1 \frac{m_2}{g}q + a_2 \frac{m_1}{g}p \pmod{LCM(m_1, m_2)}$$

Working with n Equations

When there are n equations, we simply merge two equations at a time until there is only one left. The last remaining equation is our desired answer. If at any point, if we are unable to merge two equations due to conflict, i.e, $a_i \not\equiv a_j \pmod{GCD(m_i, m_j)}$, then there is no solution.

Chinese Remainder Theorem: Strong Form

Given two sequences of numbers $A = [a_1, a_2, \ldots, a_n]$ and $M = [m_1, m_2, \ldots, m_n]$, a solution to

x exists for the following n congrunce equations:

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\dots$$
$$x \equiv a_n \pmod{m_n}$$

if, $a_i \equiv a_j \pmod{GCD(m_i, m_j)}$ and the solution will be unique modulo $L = LCM(m_1, m_2, \dots, m_n)$.

Code

The code is almost same as weak form, with slight changes in few lines.

```
/** Works for non-coprime moduli.
 Returns {-1,-1} if solution does not exist or input is invalid.
 Otherwise, returns {x,L}, where x is the solution unique to mod L
*/
pair<int, int> chinese_remainder_theorem( vector<int> A, vector<int> M ) {
    if(A.size() != M.size()) return {-1,-1}; /** Invalid input*/

    int n = A.size();

    int a1 = A[0];
    int m1 = M[0];
    /** Initially x = a_0 (mod m_0)*/

    /** Merge the solution with remaining equations */
    for ( int i = 1; i < n; i++ ) {
        int a2 = A[i];
        int m2 = M[i];

        int g = __gcd(m1, m2);
        if ( a1 % g != a2 % g ) return {-1,-1}; /** No solution exists*/

        /** Merge the two equations*/
        int p, q;
        ext_gcd(m1/g, m2/g, &p, &q);

        int mod = m1 / g * m2; /** LCM of m1 and m2*/

        /** We need to be careful about overflow, but I did not bother abo
        int x = (a1*(m2/g)*q + a2*(m1/g)*p) % mod;

        /** Merged equation*/
        a1 = x;
        if (a1 < 0) a1 += mod; /** Result is not suppose to be negative*/
        m1 = mod;
    }
    return {a1, m1};
}
```

Once again, please note that if you use `int` data type, then there is a risk of overflow. In order to keep the code simple I used `int`, but you obviously need to modify it to suit your needs. You can have a look at my CRT template on github from here: github/forthright48 - chinese_remainder_theorem.cpp

Complexity: $O(n \times log(L))$.

Conclusion

And we are done. We can now solve congruence equations even when the moduli are not pairwise coprime.

The first time I learned Chinese Remainder Theorem, I had to read a paper. The paper was well written, but I never actually managed to understand how it worked. Therefore, I always feared Chinese Remainder Theorem and used it as a black box. Until last week, I didn't even know that CRT worked with moduli that are not coprime. I really enjoyed learning how it works and all the related proof. Hopefully, you enjoyed it too.

Resources

1. forthright48 - Chinese Remainder Theorem Part 1 https://forthright48.com/2017/11/15/chinese-remainder-theorem-part-1-coprime-moduli/
2. Wiki - Chinese Remainder Theorem - https://en.wikipedia.org/wiki/Chinese_remainder_theorem
3. Brilliant.org - Chinese Remainder Theorem - https://brilliant.org/wiki/chinese-remainder-theorem/
4. Cut The Knot - Chinese Remainder Theorem - https://www.cut-the-knot.org/blue/chinese.shtml

Related Problems

1. LOJ 1319 - Monkey Tradition
2. HR - Cheese and Random Toppings
3. Kattis - generalchineseremainder

4. CF - Remainders Game

Category: CPPS, Number Theory **Tag:** problem-list, proof, theorem

Previous:
Chinese Remainder Theorem Part 1 – Coprime Moduli

Next:
Prufer Code: Linear Representation of a Labeled Tree

Archives

September 2018 (2)

February 2018 (1)

January 2018 (1)

November 2017 (2)

September 2015 (7)

August 2015 (13)

July 2015 (15)

Categories

CPPS (40)

- Combinatorics (3)
- Number Theory (33)

Meta (1)

Recent Comments

forthright48 on Number of Divisors of an Integer

Sohana Akter on Number of Divisors of an Integer

forthright48 on Number of Divisors of an Integer

forthright48 on Prufer Code: Linear Representation of a Labeled Tree

Mohammad Samiul Islam on SPOJ LCMSUM – LCM Sum

Privacy Policy