forthright48

learning never ends

# Chinese Remainder Theorem Part 1 – Coprime Moduli

forthright48 on November 15, 2017

Second part of the series can be found on: Chinese Remainder Theorem Part 2 – Non Coprime Moduli

---

Wow. It has been two years since I published my last post. Time sure flies by quickly. I have been thinking about resuming writing again for a while now. Took me long enough to get back into the mood.

I am really excited about today's post. I have been meaning to write on Chinese Remainder Theorem (CRT for short) for over two years now. Hopefully, everyone will find it interesting and easy to understand.

Here are the pre-requisite that you need to complete for understanding the post properly:

- Extended Euclidean Algorithm: Make sure you understand Bezout's Identity.

1
Shares

I remember reading the following story the first time I tried learning about CRT. It's quite interesting in my opinion. Kind of brings out the essence of CRT in a simple to understand fashion.

> **An old woman goes to market and horse steps on her basket and crashes the eggs. The rider offers to pay the damages and asks her how many eggs she had brought. She does not remember the exact number, but when she had taken them out two at a time, there was one egg left. The same happened when she picked 2,3,4,5,6 at a time, but when she took seven at a time they came out even. What is the smallest number of eggs she could have had?**

Hopefully, everyone understood the story above. Let me now rephrase it in mathematical notations.

Suppose, the old woman had $x$ eggs in her basket. Now she claims that when she took out eggs from the basket $2$ at a time, there was only $1$ egg remaining, meaning, $x \equiv 1 \pmod 2$. So basically, she is giving us various congruence equations. From her statement, we get the following linear congruence equations:

$$\begin{gathered}x \equiv 1 \pmod 2\\ x \equiv 1 \pmod 3\\ x \equiv 1 \pmod 4\\ x \equiv 1 \pmod 5\\ x \equiv 1 \pmod 6\\ x \equiv 2 \pmod 7\end{gathered}$$

Now, we need to find the smallest value of $x$ which satisfies all of the above congruence equations. Chinese Remainder Theorem helps us in finding the value of $x$.

Before we move on, let us redefine the problem formally.

## Problem: Formal Definition

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

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

## Chinese Remainder Theorem: Weak Form

As mentioned before, we can use Chinese Remainder Theorem to solve the above problem described. On this part of the CRT series, we will look into a weaker form of Chinese Remainder Theorem, which is easier to understand and occurs more frequently.

The weak Chinese Remainder Theorem states the following:

Given two sequences of numbers $A = [a_1, a_2, \dots, a_n]$ and $M = [m_1, m_2, \dots, m_n]$, where all elements of $M$ are pairwise coprime, there always exists an unique solution to $x$ mod $L$, where $L = m_1 \times m_2 \times \dots \times m_n$, such that $x$ satisfies the following linear congruence equations:

$$\begin{gathered} x \equiv a_1 (\bmod m_1) \\ x \equiv a_2 (\bmod m_2) \\ \dots \\ x \equiv a_n (\bmod m_n) \end{gathered}$$

So the weak form of Chinese Remainder Theorem has a constraint: members of the array $M$ must be pairwise coprime. What do we mean by that? This means that $\text{GCD}(m_i, m_j) = 1$ when $i \neq j$.

As long as this condition is satisfied, the weak form of CRT state that a solution to $x$ always **exists** which is "unique mod $L$". Let us see an example to understand what it means to be "unique mod $L$".

## Weak Form of CRT: Example

Suppose we are given the following three congruence equations:

$$\begin{gathered} x \equiv 3 (\bmod 5) \\ x \equiv 2 (\bmod 7) \\ x \equiv 2 (\bmod 8) \end{gathered}$$

Here we have $A = [3, 2, 2]$, $M = [5, 7, 8]$ and $L = 5 \times 7 \times 8 = 280$. Using Weak form of Chinese Remainder Theorem, we can find that $x \equiv 58 (\bmod 280)$. How did we find this? We will see that later.

Before you continue, please verify yourself that this indeed satisfies the given congruence. Also, $x = 58$ is the smallest solution and there exists more than one solution. $x = 58 + 280 \times k$, where $k$ is any non-negative integer, also satisfies the equations. But as we said before, $x$ is unique when you mod all the solutions with $L$. This is what it means to have a solution unique to mod $L$.

## Weak Form of CRT: Finding a Solution

We are first going to see how to solve when there are just two equations. Once we know how to solve for two equations, we will then generalize the method for $n$ equations.

## When there are just two equations

Let us first just consider two equation: $x \equiv a_1 \pmod{m_1}$ and $x \equiv a_2 \pmod{m_2}$. What we are going to do is merge these two equations into one equation. Here is how it works.

Since $gcd(m_1, m_2) = 1$, from Bezout's Identity, we know that there exists a solution to the following equation:

$$m_1 p + m_2 q = 1 \tag{1}$$

Using Extended Euclidean Algorithm, we can find the value of $p$ and $q$. If we know the value of $p$ and $q$, then we can say that:
$x = a_1 m_2 q + a_2 m_1 p \pmod{m_1 m_2} tag2$

See below to understand why so.

$x = a_1 m_2 q + a_2 m_1 p$
$x = a_1(1 - m_1 p) + a_2 m_1 p$ (Using (1))
$x = a_1 - a_1 m_1 p + a_2 m_1 p$
$x = a_1 + (a_2 - a_1) m_1 p$
$\therefore x \equiv a_1 \pmod{m_1}$

Similarly, $x = a_1 m_2 q + a_2 m_1 p \equiv a_2 \pmod{m_2}$

Great! We now have a possible solution for $x$, but why did we mod the solution with $m_1 m_2$? The solution we found may or may not be the smallest. Like we have seen before, there could be infinite solutions, but there exists a solution which is unique to mod $m_1 \times m_2$. How so? I guess this is the perfect time to look into its proof.

### Proof of Uniqueness

Since there could be infinite solutions, let $x_1$ and $x_2$ be two such solution. Hence we can say the following:

$x_1 \equiv a_1 \pmod{m_1}$
$x_2 \equiv a_1 \pmod{m_1}$

$x_2 \equiv a_1 (\text{mod } m_1)$
$\therefore x_1 \equiv x_2 (\text{mod } m_1)$
$x_1 - x_2 \equiv 0 (\text{mod } m_1)$
$\therefore m_1 | x_1 - x_2$

Similarly, we can show that $m_2 | x_1 - x_2$.

What can we do with all these information? The difference of any two solutions $x_1$ and $x_2$ is divisible by both $m_1$ and $m_2$. We also know that, $m_1$ and $m_2$ are coprime. Doesn't that mean, the difference is also divisible by $m_1 \times m_2$? Yes. That's exactly what we are going towards

$m_1 m_2 | x_1 - x_2$
$x_1 - x_2 \equiv 0 (\text{mod } m_1 m_2)$
$x_1 \equiv x_2 (\text{mod } m_1 m_2)$

The next question I will ask is: if the difference between any two solutions, $x_1$ and $x_2$, is divisible by $m_1 m_2$, then how many solution can there exist in the range $0$ to $m_1 m_2 - 1$? Only one. And hence, the solution $x$ is unique to modulo $m_1 m_2$.

Therefore we can say that the solution $x \equiv a_1 m_2 q + a_2 m_1 p \pmod{m_1 m_2}$ is unique and smallest. Hence, the two equation, $x \equiv a_1 \pmod{m_1}$ and $x \equiv a_2 \pmod{m_2}$, after merging becomes $x \equiv a_1 m_2 q + a_2 m_1 p \pmod{m_1 m_2}$

### When there are $n$ equations

Since we know how to merge two equations into one, all we need to do is take the first two equations from $n$ equations and merge them. Then we are left with $n-1$ equations. Since all the elements of $M$ were pairwise coprime, the new modulus is also coprime. Hence, we can continue to merge the equations until there is only one equation left. The equation left is our answer.

## Weak form of CRT: Code

The following code implements the idea we discussed above. Note that I used `int` data type, but in most of the problems you will most likely require `long long`. I am sure you can adjust the code accordingly yourself.

```
/** Return {-1,-1} if invalid input.
    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];

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

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

        /** Merged equation*/
        a1 = x;
        m1 = m1 * m2;
    }
    if (a1 < 0) a1 += m1; /** Result is not suppose to be negative*/
    return {a1, m1};
}
```

Once again, please be careful about overflow when solving a problem. From my experience, I have seen that the value of $p$ and $q$ becomes large quickly and intermediate calculations no longer fit into `long long` variables. I use `__int128` data type to avoid overflow issues. So if you get "Wrong Answer", it is most likely due to overflow.

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

## Conclusion

With this, we are done with Weak Form of Chinese Remainder Theorem. We can now find a solution to congruence equations when the moduli are co-prime. On next post, we will see a stronger version of Chinese Remainder Theorem, where the moduli are not co-prime.

**Edit:** The next post can be found here: Chinese Remainder Theorem Part 2 – Non Coprime Moduli

## Resources

1. Wiki – Chinese Remainder Theorem – https://en.wikipedia.org/wiki/Chinese_remainder_theorem

2. Brilliant.org – Chinese Remainder Theorem – https://brilliant.org/wiki/chinese-remainder-theorem/
3. 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

That's all I could find on CRT. If you know about any other related problem, please feel free to add it as a comment.

Share Tweet Share Share Email
Print Share

**Category:** CPPS, Number Theory

**Previous:**
**Modular Inverse from 1 to N**

**Next:**
**Chinese Remainder Theorem Part 2 – Non Coprime Moduli**

Guest

Your comment...

4 comments Sort by oldest

nur islam
12 months ago
on line 5 it should be A.size(); // missing the parenthesis '()'

and line 14 line should be replaced by n.
and return value {a1, m1} a1 can be negative.

Reply 0

Mohammad Samiul Islam
12 months ago

Fixed. Thank you :)

Reply 0

Shivan Nawal
2 months ago

A pretty good post but you should include the proof for equation (2) for completely understanding this theorem. I had to look up the proof elsewhere but once I understood it, everything made sense. Thanks !

Reply 0

Mohammad Samiul Islam
2 months ago

That thing has a proof? I thought some Mathematician just guessed that equation and later proved the hypothesis (they do it like this sometimes).

Can you please share where you looked up the proof? I will give it a read and add the proof here for completeness.

Reply 0

## 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