CMPSCI 250: Introduction to Computation

Lecture #14: The Chinese Remainder Theorem
David Mix Barrington
24 February 2012

The Chinese Remainder Theorem

- Infinitely Many Primes
- Reviewing Inverses and the Inverse Theorem
- Systems of Congruences, Examples
- The Simple (Two Modulus) Version
- Proving the Simple Version
- The Full (Many Modulus) Version
- Working With Really Big Numbers

Infinitely Many Primes

- There is one argument I want to squeeze in at least briefly, although its section (3.4) is not on the syllabus. How do we know that there are always more prime numbers, no matter how high in the naturals we look? We now know enough to prove this, as did the ancient Greeks.

- Let z be arbitrary -- we will prove that there exists a prime number greater than z. The **factorial** of z, written "z!", is the product of all the numbers from 1 through z (so for example $7! = 1\times2\times3\times4\times5\times6\times7 = 5040$).

- Look at the number $z! + 1$. It is not divisible by any number k in the range from 2 through z, because k must divide z! and thus $z! + 1 \equiv 1 \pmod{k}$.

- But $z! + 1$ must have a prime factorization because every positive natural does. It is either prime itself or is divisible by some smaller prime, and that prime cannot be less than or equal to z. So we know that some prime greater than z must exist, though we haven't explicitly computed it.

Reviewing Inverses and the Inverse Theorem

- We have been working with arithmetic where the “numbers” are congruence classes modulo m. A class [x] (the set {n: n ≡ x}) has a **multiplicative inverse** if there is another class [y] such that [x][y] = [1], or xy ≡ 1 (mod m).

- The **Inverse Theorem** says that a number z has a multiplicative inverse modulo m if and only if z and m are **relatively prime**, meaning that gcd(z, m) = 1. It’s fairly clear that if z and m have a common factor g > 1, then a multiplicative inverse for z modulo m is impossible.

- The **Euclidean Algorithm** is our method to compute gcd’s and thus test for relative primality. The **Extended Euclidean Algorithm** takes z and m as inputs and uses the arithmetic from the Euclidean Algorithm to write each number that occurs as an integer **linear combination** of z and m. If z and m are relatively prime, we compute numbers a and b such that az + bm = 1. Then a is an inverse of z modulo m and b is an inverse of m modulo z.

Systems of Congruences

- Modular arithmetic was invented to deal with periodic processes. We've seen how to work with multiple congruences that have the same period -- for example, we know that if $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $ab \equiv cd \pmod{m}$.

- But we sometimes have interacting periodic processes with different moduli. For example, days of the week have period 7. Suppose you have to take a pill every five days. How often do you take a pill on a Wednesday? Every 35 days, as it turns out.

- A mod-5 process and a mod-7 process interact to give a mod-35 process, and something similar happens whenever the moduli are relatively prime. If two moduli are *not* relatively prime, the two congruences may not have any common solution -- consider $x \equiv 1 \pmod{4}$ and $x \equiv 4 \pmod{6}$.

Examples of Congruence Systems

- Suppose we have around a thousand soldiers marching along the road and we would like to know exactly how many there are. We tell them to line up in rows of 7 and determine how many are left over. Then we do the same for rows of 8, then again for rows of 9. The full form of the Chinese Remainder Theorem, which we will soon prove, says that we can use these three remainders to find the number of soldiers modulo $7 \times 8 \times 9 = 504$. It might say, for example, that the number is either 806 or 1310, and hopefully we can tell the difference between these two cases.

- The pseudoscientific (i.e. “wrong”) theory of biorhythms says that a person has three cycles started at birth, of 23, 28, and 33 days. According to the same theorem, a person would be at the initial position of all three cycles again exactly $23 \times 28 \times 33 = 21252$ days, or about 58.2 years, after birth.

The Simple (Two Modulus) Version

- How can we find a common solution to the two congruences $x \equiv a \pmod{m}$ and $x \equiv b \pmod{n}$? The **Simple Version of the Chinese Remainder Theorem** says that if m and n are relatively prime, this pair of congruences is equivalent to the single congruence $x \equiv c \pmod{mn}$, where c is a number that we can calculate from a, b, m, and n.

- Note first that if x is a solution to the two congruences, so is any y that satisfies $x \equiv y \pmod{mn}$. This is because in this case $y = x + kmn$ for some integer k, and when we divide y by m, for example, we get the remainder for x plus the remainder for kmn, and the latter is 0 because m divides kmn.

- We need to find a c that gives us a solution to the two congruences, and also show that *any* solution x to the two congruences must satisfy $x \equiv c \pmod{mn}$.

Proving the Simple Version

- Since m and n are assumed to be relatively prime, the Inverse Algorithm gives us integers y and z such that ym + zn = 1.

- Our number c will be bym + azn. Let's verify that this works. When we divide bym + azn by m, the first term gives remainder 0 and the second gives [azn] = [a][zn] = [a][1] = [a]. Dividing bym + azn by n, the first term gives [b][ym] = [b][1] = [b], and the second term gives 0. A good way to think of this is that the original equation ym + zn = 1 tells us how to get a number whose remainders are 1 (mod m) and 1(mod n), and to get arbitrary a and b we can adjust either term without affecting the remainder for the other modulus.

- Let x be any solution to x ≡ a (mod m) and x ≡ b (mod n), and let d be x - c. Then d is divisible by both m and n. Use the Euclidean Algorithm to find the gcd of d and mn (or -d and mn, if d is negative) -- call this q. But q is a **common multiple** of m and n, and the least common multiple of two relatively prime numbers is their product.

The Full (Many Modulus) Version

- More generally, as in our examples, suppose we have several congruences $x = a_1 \pmod{m_1}$, $x = a_2 \pmod{m_2}$,... $x = a_k \pmod{m_k}$, and that the moduli are **pairwise relatively prime**. (This means that any two of them are relatively prime to each other.) Then the **Full Form of the Chinese Remainder Theorem** says that this system of congruences is equivalent to a single congruence $x \equiv c \pmod{M}$, where M is the product of the m_i's and c is a number that can be calculated from the a_i's and the m_i's.

- We can prove the Full Version from the Simple Version. If $k = 3$, for example, we first use the Simple Version to find a c such that the first two congruences are equivalent to $x \equiv c \pmod{m_1 m_2}$. Then we have two congruences, that and $x \equiv a_3 \pmod{m_3}$, and we can use the Simple Version again to get a common solution to them. (The pairwise relatively prime property guarantees that $m_1 m_2$ will be relatively prime to m_3.) This clearly extends to larger k. In the book, it is shown how we can calculate the single c directly.

Working With Really Big Numbers

- If I have some very very big integers, each too big to store in a single computer word, the Chinese Remainder Theorem gives me an alternate way to calculate them.

- Say I want to multiply n of these numbers together. I pick a bunch of different prime numbers, so many that their product is bigger than the product of my big numbers. (We know that such primes exist -- a more sophisticated analysis shows that there are lots of primes that fit in a single word, so I can get to very very big numbers by multiplying them together.) I then find the remainder of each big number modulo each prime.

- If I multiply together all the remainders for a given prime p, and take the result modulo p, I have my product's remainder modulo p. And this can be done with calculations on reasonably-sized numbers. I can do this in parallel for each prime. Then running the Chinese Remainder calculation once, I can get my product in the regular notation.