Number Theory II

Margaret M. Fleck

8 February 2010

This lecture covers elementary number theory concepts found in sections 3.4 and 3.5 of Rosen.

1 Announcements

Quiz this Wednesday (10th), at start of class. If you are confused about anything from HW 1 and HW 2, or other topics, don't be shy about asking questions in your discussion section. Probably several other students in your section also need help on the same topic.

Latex getting started session this Wednesday evening 7-9pm in 3405 Siebel.

Check out the newsgroup for an important small correction to homework 3.

2 Recap

Last class, we defined what it means for an integer a to divide an integer b ($a|b$): $b = an$ for some integer n. We then looked at a bunch of concepts that derive from the divides relation: prime and composite numbers, prime factorizations, gcd, and lcm.

3 The division theorem

Now, suppose that you have an integer a and an integer b that might not evenly divide a. We can still divide a by b, but we may be left with a remainder. For example, if 13 is divided by 4, the quotient is 3 and the remainder is 1. When both inputs are positive, most people agree about how to do this, but it's less obvious what's supposed to happen when one or both of the inputs is negative.

Number theorists have a very precise idea of how to compute quotients and remainders and it's specified by the following theorem.

> The Division Algorithm: For any integers a and b, where b is positive, there are unique integers q and r such that $a = bq + r$ and $0 \leq r < b$.

The name of this theorem is traditional but misleading, since neither the theorem nor its proof is an algorithm in the modern sense.

The important constraint here is that the remainder cannot be negative and must be in the range $[0, b)$. This forces the equation $a = bq + r$ to have only one solution (the "unique" qualifier in the theorem). The remainder of a divided by b is written $a \bmod b$.

Computing $a \bmod b$ for some representative inputs, we get:

- $11 \mod 3 = 2$.
- $60 \mod 7 = 4$
- $-11 \mod 3 = 1$. (The quotient is -4.)
- $0 \mod 7 = 0$

Programming languages typically contain one or more arithmetic operators similar to mod, named things like mod, modulo, rem, or %. These frequently allow negative inputs for the divisor (b) as well as the dividend a. They don't always agree with the mathematical definition even when the divisor is positive. Worse yet, in some cases, the output is implementation

dependent when either of the inputs is negative. The "modulo operation" entry on wikipedia has a nice table of what different languages do. This is sad, very sad. And it can cause hard-to-find bugs.

There are three take-home messages here. First, in this class, we will always use the number theory definition for mod. Second, always call this operation mod, to help make clear that it's the mathematical notion rather than (say) the C or Java function. Third, when programming, always check what your language does before feeding negative numbers to mod-type operators.

4 Proof of the division theorem

Proving the division theorem requires proving two separate claims. First, ("existence") that there are numbers q and r satisfying these equations. Second, ("uniqueness") that there is only one solution. Let's prove the existence part.

To do this, we need to make explicit an important property of the integers

> Well-ordering: Every non-empty set of natural numbers has a smallest element.

"Non-empty" simply means that the set actually contains something.

The well-ordering property of the integers probably seems obvious to you. However, it is important to state it explicitly, because it's not true for the real numbers. You've probably seen examples in calculus of infinite sequences of real numbers that approach a limit without reaching it. Well-ordering typically fails for such sequences. This is a key property that distinguishes the integers ("discrete") from the reals ("continuous").

Many variants of the well-ordering axiom are also true. E.g. as long as all the integers in the set are above some number b, then the set has a lowest integer. If all the integers in the set are below b, then the set must have a largest integer.

And now, to prove existence:

Proof: Let a and b be integers, where b is positive. Consider the set of integers $a - bq$, for all integers q.

If we pick any $q \leq \frac{a}{b}$, then $a - bq$ is non-negative. So our set does contain some values. Using the well-ordering property, choose the smallest non-negative value of $a - bq$. Let's call it r.

Since $r = a - bq$, then $bq + r = a$. So r satisfies the first equation from the division theorem. The way we chose r guarantees that it's ≥ 0. So we just need to show that $r < b$.

Suppose that r were $\geq b$. Consider $r - b$. $r - b$ is less than r. It's still non-negative. And it must also have the form $a - bq$. (Reduce the size of q by 1.) So this contradicts our choice of r as the smallest non-negative integer of the form $a - bq$. Therefore, it must be the case that $r < b$. $\square$.

To prove the uniqueness part of the theorem, you would pick two possible solutions q_1,r_1 and q_2,r_2 to the pair of equations. Then fiddle with the algebra to show that the two solutions must be equal.

5 Congruence mod k

If two integers a and b have the same remainder mod k, they are said to be "congruent mod k." For example, 47 and 15 are congruent mod 8, because $47 \bmod 8 = 7 = 15 \bmod 8$. You can do arithmetic on integers but only keep the remainders mod k rather than the full results of operations like addition and multiplication. This is called modular arithmetic. So, if we add 5 and 6 modulo 7, we get 11 mod 7, which is 4.

Modular arithmetic is often useful in describing periodic phenomena. For example, many facts about real-world schedules are determined by the hour of the day and the day of the week (e.g. this class meets Monday, Wednesday, and Friday). To describe these patterns, we use arithmetic mod 12 to describe the fact that hours of the day go up to 12 and then wrap around back to 1. And we can use arithmetic mod 7 to describe the fact that, within a month, days whose numbers are congruent mod 7 lie on the same day of the week.

The shorthand notation for a is congruent to b mod k is $a \equiv b \pmod{k}$. Logically, we really should write something like $a \equiv_k b$, so it looked like the notation for (say) a logarithm with a particular choice of base. In congruence mod k, k is really a modifier on our notation of equality ($\equiv$). However, $a \equiv b \pmod{k}$ has become the standard and we have to live with it.

To get used to this predicate, here are some concrete examples:

- $17 \equiv 5 \pmod{12}$ (Familiar to those of us who've had to convert between US 12-hour clocks and European or military 24-hour clocks.)
- $3 \equiv 10 \pmod{7}$
- $-3 \equiv 4 \pmod{7}$ (Since $(-3) + 7 = 4$.)
- $-3 \not\equiv 3 \pmod{7}$
- $-3 \equiv 3 \pmod{6}$
- $-21 \equiv -13 \pmod{8}$ (For both -21 and -13, the remainder mod 8 is 3.)

6 A digression on types

Argh. We're now using the notation "mod" in two related but quite distinct ways. How horribly confusing! Unfortunately, this "overloading" of the symbol mod has become standard, so you're stuck with just getting used to it.

To keep from being confused, first notice that we have recently seen two quite distinct sorts of operations on pairs of integers.

First, there are arithmetic operators, which output a number. These include functions like $+$, $*$, etc. Also gcd and lcm. And also a mod k, which returns the remainder if you divide a by k.

Second, there are predicates which output a boolean (true or false). These include $a < b$, $a = b$, $a \mid b$, and $a \equiv b \pmod{k}$.

Both in programming and mathematics, it's very important to be clear about whether a given function returns a number or a boolean, because arithmetic operations and predicates are used in very different ways.

When reading notation, you frequently have to decide whether mod is supposed to be the arithmetic operator or the boolean. Notice that the boolean version has the mod in parentheses and it always directly follows an equation using $\equiv$. When mod is used as an arithmetic operator, it's not in parentheses and there's usually no $\equiv$ nearby.

7 The official formal definition

To get a formal definition that makes proofs easy, we'll have to rephrase this idea slightly.

> Definition: If k is any positive integer, two integers a and b are congruent mod k if $k \mid (a - b)$.

This will be our official definition. It is then possible to prove that (as you might expect) that this definition is equivalent to testing whether the two numbers have the same remainder mod k. Specifically:

> Theorem: For any integers a and b and any positive integer k, $a \equiv b \pmod{k}$ if and only if $a \bmod k = b \bmod k$.

Let's try using our official definition to prove a simple fact about modular arithmetic:

Claim 1 *For any integers a, b, c, d, and k, k positive, if $a \equiv b \pmod{k}$ and $c \equiv d \pmod{k}$, then $a + c \equiv b + d \pmod{k}$.*

> Proof: Let a, b, c, d, and k be integers with k positive. Suppose that $a \equiv b \pmod{k}$ and $c \equiv d \pmod{k}$.
>
> Since $a \equiv b \pmod{k}$, $k \mid (a - b)$, by the definition of congruence mod k. Similarly, $c \equiv d \pmod{k}$, $k \mid (c - d)$.
>
> Since $k \mid (a-b)$ and $k \mid (c-d)$, we know by a lemma about divides from last week that $k \mid (a - b) + (c - d)$. So $k \mid (a + c) - (b + d)$
>
> But then the definition of congruence mod k tells us that $a + c \equiv b + d \pmod{k}$. $\square$