CS 70 Discrete Mathematics and Probability Theory
Summer 2016 Dinh, Psomas, and Ye Note 20

1 Modular Arithmetic

Suppose you go to bed at 23:00 o'clock and want to get 8 hours of sleep. What time should you set your alarm for? Clearly, the answer is not $(23+8=)$ 31:00 o'clock, but rather $(31-24=)$ 7:00 o'clock. This is because in the 24-hour clock system the numbers "wrap around" back to 0:00 once we hit 24:00, so the naive answer 31:00 means the same thing as the correct answer 7:00, namely, 7 hours after midnight.

The clock system is an example of a very useful type of arithmetic known as *modular arithmetic*, in which we perform all arithmetic operations relative to a fixed number n called the *modulus*. In the 24-hour clock system the modulus is $n = 24$, and we can write the example above as

$$31 \pmod{24} \;=\; 7$$

which is read "31 *modulo* 24 is equal to 7." Here the function "(mod 24)" is another name for remainder when divided by 24. So 31 (mod 24) is equal to 7 because $31 = 1 \cdot 24 + 7$ and $0 \le 7 \le 23$.

There are many other ways we use modular arithmetic in everyday life. As another example, consider the 7 days in a week. Suppose instead of calling them {Monday, Tuesday, ..., Sunday}, we label them with $\{0, 1, \ldots, 6\}$. Then we can describe the 7-day system with arithmetic modulo $n = 7$. For instance, the sentence "3 days after Sunday is Wednesday" can be translated into the statement

$$6+3 \pmod 7 \;=\; 2$$

and this holds because $6+3 = 9$ and when we divide 9 by 7 we get a remainder of 2. You can also see this by imagining the numbers wrapped around a circle where 7 is identified with 0 (see Figure 1), so adding or subtracting numbers correspond to walking around the circle forward or backward.

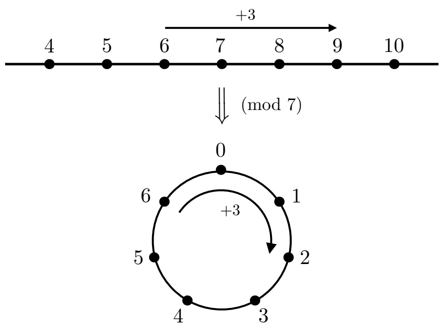

Figure 1: Addition by 3 on the integers (top) and modulo 7 (bottom).

Convince yourself that you can do the same with the 30 days in a month, 12 months in a year, etc.

So far these are simple examples from everyday life. But modular arithmetic is a very useful tool that lies at the heart of many powerful results in computer science, such as error-correcting codes and cryptography, including public-key cryptosystem such as RSA. To understand these results, let us first familiarize ourselves with modular arithmetic and explore its properties.

1.1 Addition, Subtraction, Multiplication

One of the most important properties of numbers modulo n is that the familiar arithmetic operations give robust results. The best way to illustrate this is by example. Suppose we wish to compute $42+35 \pmod{24}$. Here are two different ways of doing so. One way is to first compute the addition, then reduce $\pmod{24}$:

$$42+35 \pmod{24} \;=\; 77 \pmod{24} \;=\; 77-72 \;=\; 5.$$

The second way is to first apply $\pmod{24}$ to each term, add the results, and finally apply $\pmod{24}$ again:

$$42+35 \pmod{24} \;=\; [42 \pmod{24}]+[35 \pmod{24}] \pmod{24} \;=\; 18+11 \pmod{24} \;=\; 29 \pmod{24} \;=\; 5.$$

You see that we still get the same answer as before. This is no coincidence. We can perform any sequence of arithmetic operations $\pmod{24}$ and the result is robust — it remains unchanged whether we reduce $\pmod{24}$ only once at the end of all operations, or we reduce each intermediate result $\pmod{24}$, or even if we reduce some intermediate results $\pmod{24}$ at whim as long as we reduce the final answer $\pmod{24}$. Why is this the case? To see this we must view numbers modulo n in a completely different way.

For two integers a and b, let us define a to be *congruent* to b modulo n, written

$$a \;\equiv_n\; b$$

if and only if $a \pmod{n} = b \pmod{n}$, i.e., a and b give the same remainder when divided by n. In the above scenario, this captures the property that a and b would result in the same answer when used in the arithmetic operations. Here is another way of saying the same thing:

$$a \equiv_n b \;\;\Leftrightarrow\;\; n \,|\, (a-b)$$

(recall that $p\,|\,q$ means p divides q, i.e., q/p is an integer). Equivalently, we can write $a = b+k\cdot n$ for some integer k. Make sure you understand and can show that these conditions are all saying the same thing.

So when $n = 24$, all these numbers are congruent to each other:

$$\ldots \;\equiv_{24}\; -48 \;\equiv_{24}\; -24 \;\equiv_{24}\; 0 \;\equiv_{24}\; 24 \;\equiv_{24}\; 48 \;\equiv_{24}\; \ldots$$

Similarly,

$$\ldots \;\equiv_{24}\; -47 \;\equiv_{24}\; -23 \;\equiv_{24}\; 1 \;\equiv_{24}\; 25 \;\equiv_{24}\; 49 \;\equiv_{24}\; \cdots$$

and so on. Indeed, you can see that there are 24 such sets where the numbers within each set are all congruent to each other.

The reason that arithmetic modulo n is robust is that all the arithmetic operations — plus, minus, times — respect these sets. This is proved in the following theorem:

Theorem 20.1. *For all $n \geq 1$ and $a,b,c,d \in \mathbb{Z}$, the following are true:*

1. *If $a \equiv_n b$ and $c \equiv_n d$, then $a+c \equiv_n b+d$*
2. *If $a \equiv_n b$ and $c \equiv_n d$, then $a\cdot c \equiv_n b\cdot d$*

Proof. We prove part (1). The proof of (2) is left as an exercise.

Since $a \equiv_n b$, there exists $k \in \mathbb{Z}$ such that $a = b+k\cdot n$. Similarly, since $c \equiv_n d$, there exists $\ell \in \mathbb{Z}$ such that $c = d+\ell\cdot n$. Then we can write

$$a+c = (b+d) \,+\, (k+\ell)\cdot n$$

which means $(a+c)-(b+d)$ is divisible by n. This shows that $a+c \equiv_n b+d$. □

Remark: In our notation so far, $\pmod n$ is a function that maps an integer a to an element in the set $\{0,1,\ldots,n-1\}$ by computing the remainder when dividing a by n. The notation $\equiv_n$ says that two integers give the same remainder when divided by n. These two concepts are related, but quite different from each other. Once we get used to the two concepts, it is customary to blur the distinction between them and use the notation $\pmod n$ interchangeably: to denote the remainder, but also to denote equivalence as in

$$a \equiv b \pmod n$$

to denote $a \equiv_n b$.

2 Division and Multiplicative Inverses

In Section 1.1 we discussed addition, subtraction, and multiplication in the setting of modular arithmetic. Curiously, we left out division. The reason is that division is a bit more tricky in this setting, and thus warrants its own discussion. To begin, let's recall how division is done over the rational numbers $\mathbb{Q}$. Here, if we want to divide $x \in \mathbb{Q}$ by $y \in \mathbb{Q}$, we can equivalently multiply by y^{-1}, where if $y = a/b$, then $y^{-1} = b/a$. The term y^{-1} has a special name — it is a *multiplicative inverse* of y, i.e. a number such that $y \cdot y^{-1} = 1$. This suggests the following approach for dividing number x by y in modular arithmetic: Simply multiply x by y^{-1}.

But what is the multiplicative inverse of y modulo n? Is it unique as in the case of working over the rationals? Even more troubling — is it clear the inverse even *exists*? Remarkably, both scenarios are possible in modular arithmetic: Sometimes the inverse doesn't exist, and sometimes it does. When it *does* exist, however, it turns out to be unique. Let's explore these ideas further.

Consider the case of $x = 8$. Does x have a multiplicative inverse modulo $n = 15$? Yes! Note that $2x \equiv 16 \equiv 1 \pmod{15}$. Thus, 2 is a multiplicative inverse of x modulo 15.

Sanity check! Consider now $x = 12$ and $n = 15$. Does x have a multiplicative inverse modulo m? (Hint: Since we are working modulo 15, there are only 15 possible choices for the inverse. Try plugging in $a = 0,1,2,\ldots$ into $ax \pmod n$ and seeing if you get the answer 1. Do you see a pattern forming?)

So sometimes the inverse exists, and sometimes it does not. Is there a way to distinguish between the two cases? Yes — it turns out that x has a multiplicative inverse modulo n if and only if the greatest common divisor of n and x is 1. Here, the *greatest common divisor* of two natural numbers x and y, denoted $\gcd(x,y)$, is the largest natural number that divides them both. For example, $\gcd(30,24) = 6$. If $\gcd(x,y) = 1$, it means that x and y share no common factors (except 1); in this case we call x and y *relatively prime*.

Sanity check! What is $\gcd(21,49)$? How about $\gcd(12,13)$?

We can show that the multiplicative inverse exists precisely when the number x is relatively prime to the modulus n. The proof is left as an exercise.

Theorem 20.2. *Let n,x be positive integers. Then x has a multiplicative inverse modulo n if and only if* $\gcd(n,x) = 1$. *Moreover, if it exists, then the multiplicative inverse is unique.*

2.1 Computing the Multiplicative Inverse

Theorem 20.2 gives us a sufficient condition for determining if a multiplicative inverse of x modulo n *exists*. But it doesn't tell us how to *find* such an inverse! In this section, we discuss an efficient algorithm for computing the inverse itself whenever it exists. Interestingly, this task is closely related to computing the greatest common divisor of two numbers, and both go via variants of *Euclid's algorithm*. It is worth noting that the latter is one of the oldest algorithms still in use, dating back to around 300 BC!

To begin, let us see how computing the multiplicative inverse of x modulo m is related to finding $\gcd(x,m)$. Suppose that for any pair of numbers x,y, we can not only compute $\gcd(x,y)$, but also a pair of integers a,b satisfying

$$d = \gcd(x,y) = ax+by. \tag{1}$$

(Note that this is not a modular equation, and the integers a,b can be zero or negative.) For example, we can write $1 = \gcd(35,12) = -1\cdot 35+3\cdot 12$, so here $a=-1$ and $b=3$ are possible values for a,b.

If we can do this, then we can compute the multiplicative inverse of x modulo n as follows. First, compute integers a and b such that

$$1 = \gcd(n,x) = an+bx.$$

This implies, however, that $bx \equiv 1 \pmod{n}$. Thus, b must be the multiplicative inverse of x modulo n! Reducing b modulo n hence gives us the *unique* inverse we are looking for.

Sanity check! In the example $1 = \gcd(35,12) = -1\cdot 35+3\cdot 12$, what is the unique multiplicative inverse 12^{-1} of 12? Verify your answer by explicitly checking that indeed $12\cdot 12^{-1} \equiv 1 \pmod{35}$.

We conclude that we have reduced the problem of computing inverses to the problem of finding integers a,b satisfying Equation (1). Remarkably, Euclid's algorithm for computing greatest common divisor *also* allows us to find the integers a and b described above, which we shall see now.

2.1.1 Euclid's Algorithm

We begin by describing the "basic" version of Euclid's algorithm, which only computes the gcd, and not the integers a and b as described in Equation (1). In the next section, we describe the *extended* version of Euclid's algorithm, which completes both tasks, as promised.

To begin, suppose we wish to compute the gcd of two numbers x and y. An easy "base case" is when either x or y is 0; for example, if $x=0$, then $\gcd(x,y)=y$, since 0 is divisible by everything. In a nutshell, Euclid's algorithm works by reducing the general case of computing $\gcd(x,y)$ down to this base case. To accomplish this, we need the following theorem that allows us to reduce the size of the integers that we are working with, thereby getting us closer to the base case.

Theorem 20.3. *Let $x \geq y$ and let q,r be natural numbers such $x = yq+r$ and $r<y$. Then* $\gcd(x,y) = \gcd(y,r)$.

Proof. We give a direct proof. Note that the claim will follow if we can show that any common divisor d of x and y is also a common divisor of r and y, and vice versa. To show the forward direction, suppose $d \mid x$ and $d \mid y$. This means $z = x/d$ and $w = y/d$ are integers. Then we see that

$$r = x-yq = zd-wdq = (z-wq)d$$

implying $d \mid r$, as desired. The converse direction follows similarly. $\square$

Exercise. Show the converse direction of the proof of Theorem 20.3, i.e., show that $d \mid y$ and $d \mid r$ imply $d \mid x$.

Given this theorem, let's see how to compute $\gcd(16,10)$. We begin by writing

$$16 = 10 \times 1 + 6.$$

Here $x = 16$, $y = 10$, $q = 1$, and $r = 6$. By Theorem 20.3, we know that $\gcd(16,10) = \gcd(10,6)$. Therefore, we can simplify the problem by now focusing on $\gcd(10,6)$, and writing

$$10 = 6 \times 1 + 4.$$

As before, Theorem 20.3 says $\gcd(10,6) = \gcd(6,4)$. You can probably see a pattern forming now. We apply the same idea on $\gcd(6,4)$, and so forth, obtaining the sequence of equations:

$$\begin{aligned} 6 &= 4 \times 1 + 2 \\ 4 &= 2 \times 2 + 0 \\ 2 &= 0 \times 0 + 2. \end{aligned}$$

Note that the last line is precisely the base case we discussed at the start of this section — we are looking for $\gcd(2,0)$, which is trivially 2. We conclude that $\gcd(16,10) = 2$, as desired.

Sanity check! Apply the algorithm sketched above to compute $\gcd(24,12)$.

Now that you've had a chance to get your hands dirty by trying out the algorithm, let's describe it formally using recursion as follows.

```
// precondition:  Assumes $x \geq y \geq 0$ and $x > 0$.
// postcondition:  Outputs $\gcd(x,y)$.
algorithm  $\gcd(x,y)$
  if $y = 0$ then return $x$
  else return $\gcd(y, x \pmod{y})$
```

You should run through a second quick example to convince yourself that this formal description really does capture the algorithm we sketched previously.

Sanity check! Compute $\gcd(32,10)$ using the formal algorithm above.

2.1.2 Extended Euclid's Algorithm

We finally give an extended version of Euclid's algorithm which computes $\gcd(x,y)$, along with integers a and b satisfying $\gcd(x,y) = ax + by$, allowing us to finally compute multiplicative inverses. The algorithm is as follows.

```
//precondition: Assumes $x \geq y \geq 0$ and $x > 0$.
//postcondition: Outputs $(d,a,b)$ where $d = \gcd(x,y)$ and $a,b \in \mathbb{Z}$ with $d = ax+by$.
algorithm extended-gcd$(x,y)$
  if $y = 0$ then return $(x,1,0)$
  else
    let $(d,a,b) :=$ extended-gcd$(y, x \pmod{y})$
    return $(d, b, a - \lfloor x/y \rfloor b)$
```

Here the *floor* $\lfloor z \rfloor$ is the largest integer c such that $c \leq z$.

Note that the extended gcd algorithm has the same form as the basic gcd algorithm we saw earlier; the only difference is that we now also carry around the values a,b. Let's make sure you believe this algorithm does what we claim via an example.

Sanity check! Run extended-gcd(x,y) on inputs $x = 16$ and $y = 10$. Check that the resulting values of a and b indeed satisfy the postcondition that $d = ax+by$.

Finally, to understand *why* this algorithm works, let us prove its correctness!

Theorem 20.4. *If x and y satisfy the preconditions of extended-gcd, then the output (d,a,b) of extended-gcd(x,y) satisfy its postconditions.*

Proof. We proceed by strong induction on y. The base case is $y = 0$, in which case the algorithm returns $(d,a,b) = (x,1,0)$. This is correct since $x = \gcd(x,0)$ and $x = 1 \cdot x + 0 \cdot y$, as desired. Assume now the claim is true for all $0 < y \leq k$. We prove the claim for $y = k+1$. Let x and y be the input to extended-gcd. Then by the induction hypothesis, the first line in the `else` clause returns (d,a,b) such that $d = \gcd(y, x \pmod{y})$ and $d = ay + b(x \pmod{y})$. Thus, our job is to check that the last line of the algorithm returns values satisfying the postcondition, i.e., that $d = \gcd(x,y)$ and that $d = bx + (a - \lfloor x/y \rfloor b)y$. The first of these is correct by Theorem 20.3. To see the second, note that we can write $x \pmod{y} = x - \lfloor x/y \rfloor y$ (check this!), so

$$d = ay + b(x \pmod{y}) = ay + b(x - \lfloor x/y \rfloor y) = bx + (a - \lfloor x/y \rfloor b)y.$$

This concludes the proof. □

3 Practice Problems

1. Complete the proof of Theorem 20.1.
2. Prove Theorem 20.2.
3. Prove that the product of any $k \geq 1$ consecutive integers is divisible by k.