Reading Assignment 2
Lecture 5

Read the content below about Modular Arithmetic.

Modular Arithmetic

Another application of the division algorithm that will be important to us is modular arithmetic. Modular arithmetic is an abstraction of a method of counting that you often use. For example, if it is now September, what month will it be 25 months from now? Of course, the answer is October, but the interesting fact is that you didn't arrive at the answer by starting with September and counting off 25 months. Instead, without even thinking about it, you simply observed that $25 = 2 \cdot 12 + 1$, and you added 1 month to September. Similarly, if it is now Wednesday, you know that in 23 days it will be Friday. This time, you arrived at your answer by noting that $23 = 7 \cdot 3 + 2$, so you added 2 days to Wednesday instead of counting off 23 days. If your electricity is off for 26 hours, you must advance your clock 2 hours, since $26 = 2 \cdot 12 + 2$. Surprisingly, this simple idea has numerous important applications in mathematics and computer science. You will see a few of them in this section. We shall see many more in later chapters.

The following notation is convenient. When $a = qn + r$, where q is the quotient and r is the remainder upon dividing a by n, we write $a \bmod n = r$. Thus,

$$\begin{aligned} 3 \bmod 2 &= 1 \text{ since } 3 = 1 \cdot 2 + 1, \\ 6 \bmod 2 &= 0 \text{ since } 6 = 3 \cdot 2 + 0, \\ 11 \bmod 3 &= 2 \text{ since } 11 = 3 \cdot 3 + 2, \\ 62 \bmod 85 &= 62 \text{ since } 62 = 0 \cdot 85 + 62, \\ -2 \bmod 15 &= 13 \text{ since } -2 = (-1)15 + 13. \end{aligned}$$

In general, if a and b are integers and n is a positive integer, then $a \bmod n = b \bmod n$ if and only if n divides $a - b$ (Exercise 9).

In our applications, we will use addition and multiplication mod n. When you wish to compute ab mod n or $(a + b)$ mod n, and a or b is greater than n, it is easier to "mod first." For example, to compute $(27 \cdot 36)$ mod 11, we note that 27 mod 11 = 5 and 36 mod 11 = 3, so $(27 \cdot 36)$ mod 11 = $(5 \cdot 3)$ mod 11 = 4. (See Exercise 11.)

Modular arithmetic is often used in assigning an extra digit to identification numbers for the purpose of detecting forgery or errors. We present two such applications.

EXAMPLE 6 The United States Postal Service money order shown in Figure 0.1 has an identification number consisting of 10 digits together with an extra digit called a *check*. The check digit is the 10-digit number modulo 9. Thus, the number 3953988164 has the check digit 2, since 3953988164 mod 9 = 2.[1] If the number 39539881642 were incorrectly entered into a computer (programmed to calculate the check digit) as, say, 39559881642 (an error in the fourth position), the machine would calculate the check digit as 4, whereas the entered check digit would be 2. Thus, the error would be detected.

The method used by the Postal Service does not detect all single-digit errors (see Exercise 41). However, detection of all

[1]The value of N mod 9 is easy to compute with a calculator. If $N = 9q + r$, where r is the remainder upon dividing N by 9, then on a calculator screen $N \div 9$ appears as $q.rrrrr\ldots$, so the first decimal digit is the check digit. For example, $3953988164 \div 9 = 439332018.222$, so 2 is the check digit. If N has too many digits for your calculator, replace N by the sum of its digits and divide that number by 9. Thus, 3953988164 mod 9 = 56 mod 9 = 2. The value of 3953988164 mod 9 can also be computed by searching Google for "3953988164 mod 9."

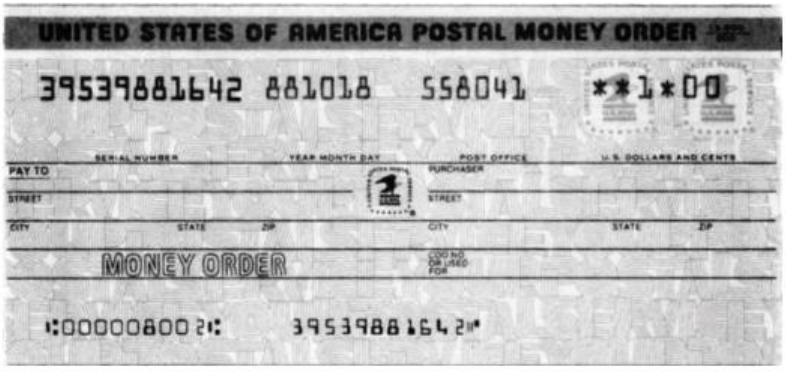

Figure 0.1 Postal money order.

single-digit errors, as well as nearly all errors involving the transposition of two adjacent digits, is easily achieved. One method that does this is the one used to assign the so-called Universal Product Code (UPC) to most retail items (see Figure 0.2). A UPC identification number has 12 digits. The first six digits identify the manufacturer, the next five identify the product, and the last is a check. (For many items, the 12th digit is not printed, but it is always bar-coded.) In Figure 0.2, the check digit is 8.

Figure 0.2 UPC bar code.

To explain how the check digit is calculated, it is convenient to introduce the dot product notation for two k-tuples:

$$(a_1, a_2, \ldots, a_k) \cdot (w_1, w_2, \ldots, w_k) = a_1w_1 + a_2w_2 + \cdots + a_kw_k.$$

An item with the UPC identification number $a_1a_2 \cdots a_{12}$ satisfies the condition

$$(a_1, a_2, \ldots, a_{12}) \cdot (3, 1, 3, 1, \ldots, 3, 1) \bmod 10 = 0.$$

To verify that the number in Figure 0.2 satisfies this condition, we calculate

$$(0 \cdot 3 + 2 \cdot 1 + 1 \cdot 3 + 0 \cdot 1 + 0 \cdot 3 + 0 \cdot 1 + 6 \cdot 3 + 5 \cdot 1$$

$$+ 8 \cdot 3 + 9 \cdot 1 + 7 \cdot 3 + 8 \cdot 1) \bmod 10 = 90 \bmod 10 = 0.$$

The fixed k-tuple used in the calculation of check digits is called the *weighting vector*.

Now suppose a single error is made in entering the number in Figure 0.2 into a computer. Say, for instance, that 021000958978 is entered (notice that the seventh digit is incorrect). Then the computer calculates

$$0 \cdot 3 + 2 \cdot 1 + 1 \cdot 3 + 0 \cdot 1 + 0 \cdot 3 + 0 \cdot 1 + 9 \cdot 3 + 5 \cdot 1 + 8 \cdot 3 + 9 \cdot 1 + 7 \cdot 3 + 8 \cdot 1 = 99.$$

Since $99 \bmod 10 \neq 0$, the entered number cannot be correct.

In general, any single error will result in a sum that is not 0 modulo 10.

The advantage of the UPC scheme is that it will detect nearly all errors involving the transposition of two adjacent digits as well as all errors involving one digit. For doubters, let us say that the identification number given in Figure 0.2 is entered as 021000658798. Notice that the last two digits preceding the check digit have been transposed. But by calculating the dot product, we obtain $94 \bmod 10 \neq 0$, so we have detected an error. In fact, the only undetected transposition errors of adjacent digits a and b are those where $|a - b| = 5$. To verify this, we observe that a transposition error of the form

$$a_1 a_2 \cdots a_i a_{i+1} \cdots a_{12} \rightarrow a_1 a_2 \cdots a_{i+1} a_i \cdots a_{12}$$

is undetected if and only if

$$(a_1, a_2, \ldots, a_{i+1}, a_i, \ldots, a_{12}) \cdot (3, 1, 3, 1, \ldots, 3, 1) \bmod 10 = 0.$$

That is, the error is undetected if and only if

$$\begin{aligned}&(a_1, a_2, \ldots, a_{i+1}, a_i, \ldots, a_{12}) \cdot (3, 1, 3, 1, \ldots, 3, 1) \bmod 10 \\ &= (a_1, a_2, \ldots, a_i, a_{i+1}, \ldots, a_{12}) \cdot (3, 1, 3, 1, \ldots, 3, 1) \bmod 10.\end{aligned}$$

This equality simplifies to either

$$(3a_{i+1} + a_i) \bmod 10 = (3a_i + a_{i+1}) \bmod 10$$

or

$$(a_{i+1} + 3a_i) \bmod 10 = (a_i + 3a_{i+1}) \bmod 10,$$

depending on whether i is even or odd. Both cases reduce to $2(a_{i+1} - a_i) \mod 10 = 0$. It follows that $|a_{i+1} - a_i| = 5$, if $a_{i+1} \neq a_i$.

In 2005, United States companies began to phase in the use of a 13th digit to be in conformance with the 13-digit product identification numbers used in Europe. The weighting vector for 13-digit numbers is $(1, 3, 1, 3, \ldots, 3, 1)$.

Identification numbers printed on bank checks (on the bottom left between the two colons) consist of an eight-digit number $a_1 a_2 \cdots a_8$ and a check digit a_9, so that

$$(a_1, a_2, \ldots, a_9) \cdot (7, 3, 9, 7, 3, 9, 7, 3, 9) \bmod 10 = 0.$$

As is the case for the UPC scheme, this method detects all single-digit errors and all errors involving the transposition of adjacent digits a and b except when $|a - b| = 5$. But it also detects most errors of the form $\cdots abc \cdots \to \cdots cba \cdots$, whereas the UPC method detects no errors of this form.

Modular arithmetic is often used to verify the validity of statements about divisibility regarding all positive integers by checking only finitely many cases.