

Lecture 2. The Euclidean Algorithm and Numbers in Other Bases

At the end of Lecture 1, we gave formulas for the greatest common divisor $GCD(a, b)$, and the least common multiple $LCM(a, b)$ of two integers a and b in terms of their prime factorizations. Unfortunately, finding the prime factorizations may be very complicated, especially if a and b are large. Now we present an alternative method for finding $GCD(a, b)$ called the *Euclidean algorithm*. It is probably the oldest example of an algorithm in mathematics (3rd century BC). It gives the additional information about so-called *linear Diophant's equations*. Diophantus, often known as 'the father of algebra', is best known for his *Arithmetica*, a work on the solution of algebraic equations and on the theory of numbers. However, essentially nothing is known of his life and there has been much debate regarding the dates during which he lived. Suppose that $a > b > 0$. Then we may write

$$a = bq_1 + r_1, \quad 0 \leq r_1 < b \quad (1)$$

(q_1 is a quotient, r_1 is a remainder).

If $r_1 = 0$, then $b|a$ and $GCD(a, b) = b$.

If $r_1 \neq 0$, then it is easy to see $GCD(a, b) = GCD(b, r_1)$.

Indeed, $d|a, d|b \implies d|r_1 = a - bq_1$, and $d'|b, d'|r_1 \implies d'|a$.

Numbers b and r_1 are less than a and b . We can continue this process

$$b = r_1q_2 + r_2 \quad (2)$$

$$r_1 = r_2q_3 + r_3 \quad (3)$$

and eventually,

$$r_{k-2} = r_{k-1}q_k + r_k$$

$$r_{k-1} = r_k \cdot q_{k+1}.$$

(Last division without remainder). It is clear that $r_k = d_0 = GCD(a, b)$.

Numerical illustration. $a = 1001$, $b = 65$ (one can use a calculator for division with remainder):

1) $1001 = 65 \cdot 15 + 26$, so $r_1 = 26$ and $q_1 = 15$.

2) $65 = 26 \cdot 2 + 13$, so $r_2 = 13$ and $q_2 = 2$.

3) $26 = 13 \cdot 2$, so $r_3 = GCD(1001, 65) = 13$.

Let's now manipulate the Euclidean algorithm starting from the very beginning. The first relation

$$a = bq_1 + r_1$$

gives

$$r_1 = a - bq_1$$

that is, the first remainder r_1 is a linear combination of the initial numbers a and b with integer coefficients. It follows from the second relation that

$$r_2 = b - r_1q_2 = b - (a - bq_1)q_2 = b(1 + q_1q_2) - aq_2$$

that is, r_2 is also a linear combination of a and b . One can continue this process to the very end, when we'll find that

$$r_k = \text{GCD}(a, b) = ax_0 + by_0$$

for appropriate integers x_0, y_0 (one of them is positive and another is negative).

Equations of the form

$$ax + by = c$$

for fixed integers a, b, c with two (!) unknown integer variables x, y are known as linear Diophantine equations.

Of course they always have an infinite number of real solutions: for arbitrary real y

$$x = \frac{c - by}{a}$$

But such an equation may have no integer solutions. For instance,

$$2x + 4y = 5$$

has none, because the left side of the equation is even for arbitrary integers x, y while the right side is odd. In general, if $d_0 = \text{GCD}(a, b)$ and $d_0 \nmid c$, then the equation $ax + by = c$ has no solutions. At the same time, if $d_0 \mid c$, then at least one solution exists. In fact, we know the equation

$$ax + by = d_0$$

is solvable by the Euclidean algorithm. Suppose that

$$ax_0 + by_0 = d_0.$$

Now $c = d_0 \cdot q \implies ax_0q + by_0q = d_0q = c$ and

$$x_1 = x_0q, y_1 = y_0q,$$

which gives the solution

$$ax + by = c$$

There is a concrete way to look at the Euclidean algorithm. It solves the *decanting problem* for two decanters. Here's the problem. We are given two ungraduated decanters (= containers) and an unlimited supply of water. Three types of operations (moves) are allowed:

1. An empty decanter can be filled with water,
2. A full decanter can be dumped out, and
3. Water can be poured from one decanter to the other until either the contributing decanter is empty or the receiving container is full.

Both decanters have integer capacities. The question is ‘what is the least amount of water that can be measured, and how can that be accomplished?’ Before considering the general question, we examine two examples. Suppose the containers have capacities 5 units and 7 units. A little experimentation leads to the conclusion that 1 unit of water can be obtained by filling the 5 unit container 3 times, pouring repeatedly from the 5 unit to the 7 unit container and dumping out the 7 unit container twice. A finite state diagram is helpful to follow the procedure:

$$(0, 0) \implies (5, 0) \implies (0, 5) \implies (5, 5) \implies (3, 7) \implies (3, 0) \implies (0, 3) \implies (5, 3) \\ (5, 3) \implies (1, 7) \implies (1, 0)$$

Notice that the procedure includes 3 fills(f) and 2 dumps(d), with fills and dumps alternating and separated by pours(p). An arithmetic equation representing this is

$$3 \cdot 5 - 2 \cdot 7 = 1.$$

Notice that not only does the arithmetic equation follow from the state diagram, the reverse is also true. That is, given the arithmetic equation it is an easy matter to construct the state diagram.

In the next example, the least amount that can be measured is not 1. Let the containers have sizes 15 and 21. Before reading on, can you see why it is impossible to obtain exactly one unit of water? An equation can be obtained for any sequence of moves. Such an equation is of the form

$$15x + 21y = z$$

where exactly one of the integers x and y is negative, and z is the amount obtained. Now notice that the left side is a multiple of 3, so the right side must be also. Thus the least amount that can be measured is 3 units. One can also reason this as follows: each fill adds a multiple of 3 units of water, each pour leaves the number unchanged, and each dump removes a multiple of three units, so the amount on hand at each stage is a multiple of 3.

In fact, the answer is that the least amount that can be measured is the greatest common divisor of the two container sizes, and the Euclidean algorithm tells us how

to proceed. Suppose containers of sizes a and b are given, and $c = GCD(a, b)$. The Euclidean algorithm yields a solution to

$$c = ax + by$$

where x and y are integers exactly one of which is positive and, except in trivial cases, the other is negative. For convenience, we assume x is positive. Then the solution to the decanting problem is to fill the a capacity container x times, repeatedly pouring its contents into the b unit container. The b unit container will be dumped y times, so the total water on hand at the end is the difference $ax - by = c$.

Historical(!) Interlude. In one of the Diehard movies, Bruce Willis is confronted with a bomb that he must diffuse. In order to accomplish this, he must balance a pressure plate with the weight of exactly 4 gallons of water which he must measure out using just a 3-gallon and a 5-gallon container. He has a fountain with enough water, and can pour and dump just as we allow in the decanting problem. Solve this problem for Bruce and diffuse the bomb.

Example 1. Lets look at another specific example. Again we use the Euclidean Algorithm to solve the decanting problem. As you know there are two stages. The first stage is a sequence of divisions. The second is a sequence of ‘unwindings’. For this example let the decanter sizes be $a = 257$ and $b = 341$ Use the division algorithm to get $341 = 1 \cdot 257 + 84$. Then apply the division algorithm again on 257 and 84 to write $257 = 3 \cdot 84 + 5$. Applying it yet again to 84 and 5 yields $84 = 16 \cdot 5 + 4$. Finally, divide 5 by 4 to get $5 = 1 \cdot 4 + 1$. This completes the first stage. Now to unwind, write $1 = 5 - 1 \cdot 4$. Then replace the 4 with $84 - 16 \cdot 5$ to get $1 = 5 - 1(84 - 16 \cdot 5)$. This is equivalent to $1 = 17 \cdot 5 - 1 \cdot 84$. Check this to be sure. Then replace 5 with $257 - 3 \cdot 84$ to get

$$1 = 17 \cdot (257 - 3 \cdot 84) - 1 \cdot 84,$$

i.e., $1 = 17 \cdot 257 - 52 \cdot 84$. Finally, replace 84 with $341 - 257$ to get $1 = 17 \cdot 257 - 52(341 - 257)$, which we can rewrite as

$$1 = 69 \cdot 257 - 52 \cdot 341.$$

Thus, the solution to the decanting problem is to measure out 1 unit of water by filling the 257 unit container 69 times, repeatedly pouring its contents into the 341 unit container, and, in the process, dumping out the 341 unit container 52 times.

Example 2. Is it possible to have 100 coins made up of x pennies, y dimes and z quarters be worth exactly \$5?

If it is possible, then

$$\begin{aligned} x + y + z &= 100, \\ x + 10y + 25z &= 500. \end{aligned}$$

This is a linear Diophantine system of two equations with three (integer and non-negative variables). Subtract the first equation from the second. We get

$$9y + 24z = 400$$

But $GCD(9, 24) = 3$ and $3 \nmid 400$. The equation

$$9y + 24z = 400,$$

as well as the initial system, has no solutions.

Example 3. The same problem as example 2, but the total value of the coins is \$7.

We have

$$\begin{aligned} x + y + z &= 100, \\ x + 10y + 25z &= 700, \end{aligned}$$

and

$$9y + 24z = 600,$$

or

$$3y + 8z = 200 \implies 3y = 200 - 8z = 8(25 - z).$$

The last equation makes it clear that y is a multiple of 8. Letting $y = 0, 8, 16, 24, \dots, 64$ yields $z = 25, 22, 19, \dots, 1$. So the system has 9 different solutions

$$(x, y, z) = (75 - 5t, 8t, 25 - 3t) \text{ for } t = 0, 1, 2, \dots, 8.$$

Now we'll describe another application of the division algorithm: numerical system with different bases. Modern computers work mainly with bases $b = 2$, $b = 4$, $b = 16$. Let's start with the simplest base $b = 2$ (binary representation).

Theorem 1. Every positive integer $n > 0$ can be represented in exactly one way as a sum of distinct, non-negative integer powers of 2.

$$n = (a_k \dots a_0)_2 = a_k \cdot 2^k + a_{k-1} \cdot 2^{k-1} + \dots + a_1 \cdot 2 + a_0$$

where

$$a_k = 1, a_{k-1} = 0 \text{ or } 1, \dots, a_0 = 0 \text{ or } 1.$$

The uniqueness theorem is not too hard to prove by contradiction. Suppose that

$$\begin{aligned} n &= a_k \cdot 2^k + \dots + a_1 \cdot 2 + a_0 \text{ and} \\ n &= b_l \cdot 2^l + \dots + b_1 \cdot 2 + b_0. \end{aligned}$$

Then

$$0 = (a_0 - b_0) + 2(a_1 - b_1) + 2^2(a_2 - b_2) + \dots$$

Possible values of $(a_0 - b_0)$ are $0, \pm 1$. At the same time, the other terms and the side of the equation are divisible by 2. This means that $a_0 - b_0 = 0$ and hence that $a_0 = b_0$. But then

$$0 = 2(a_1 - b_1) + 2^2(a_2 - b_2) + \dots$$

and division by 2 yields

$$0 = (a_1 - b_1) + 2(a_2 - b_2) + \dots$$

The same analysis gives $a_1 = b_1$ etc.

In the next few paragraphs, we'll discuss two methods for determining the representation of integers in bases other than base 10 and then two methods for representing fractions in other bases. The two integer algorithms are called *repeated subtraction* and *repeated division*. The analogous algorithms for fractions are called repeated subtraction and repeated multiplication. First we present algorithms specific to base 2.

Repeated Subtraction. Let 2^k be the largest power of 2 that is less than or equal to n , that is, $2^k \leq n < 2^{k+1}$. Then we assign the numeral a_k the value 1. Next let $n_1 = n - 2^k$. For n_1 we can continue this process to get k_1 such that

$$2^{k_1} \leq n_1 < 2^{k_1+1}$$

etc. We eventually have the representation

$$n = 2^k + 2^{k_1} + \dots$$

The binary digits $a_i, i = k, k_1, \dots, 0$ are equal to 1 for the indexes k, k_1, \dots and 0 for other indices.

Example 4. Let $n = 174$. Then

$$\begin{aligned} n &= 128 + 46 = 128 + 32 + 14 = 128 + 32 + 8 + 4 + 2 = \\ &= 2^7 + 2^5 + 2^3 + 2^2 + 2 = (10101110)_2. \end{aligned}$$

More generally, suppose we want to find the base b representation of an integer N . Find the highest power of b that is less than or equal to N , and subtract it repeatedly until the result is less than that highest power. Then repeat the process on the remainder. For example, suppose we want the base 5 representation of 89. The highest power of 5 that does not exceed 89 is $5^2 = 25$ repeatedly subtracting 25 from 89 yields $89 = 25 + 63 = 2 \cdot 25 + 39 = 3 \cdot 25 + 14$. The highest power of

5 not bigger than 14 is 5 itself, so repeatedly subtract 5 from the remainder 14 to get $89 = 3 \cdot 5^2 + 2 \cdot 5 + 4$. Now we can see how to write 89 as a sum of multiples of powers of 5:

$$89 = 3 \cdot 5^2 + 2 \cdot 5^1 + 4 \cdot 5^0 = 324_5.$$

This means that the base 5 representation of 89 is 324_5 .

Repeated Division. The second algorithm, called Repeated Division, is harder to understand, but much easier and quicker to carry out. It is based on the following operations:

$$\begin{aligned} n &= 2n_1 + a_0 \\ n_1 &= 2n_2 + a_1 \\ &\vdots \\ n_{k-1} &= 2n_k + a_{k-1} \\ n_k &= 1. \end{aligned}$$

$$\begin{array}{ll} 174 = 2 \cdot 87 + 0 & a_0 = 0 \\ 87 = 2 \cdot 43 + 1 & a_1 = 1 \\ 43 = 2 \cdot 21 + 1 & a_2 = 1 \\ 21 = 2 \cdot 10 + 1 & a_3 = 1 \\ 10 = 2 \cdot 5 + 0 & a_4 = 0 \\ 5 = 2 \cdot 2 + 1 & a_5 = 1 \\ 2 = 2 \cdot 1 + 0 & a_6 = 0 \\ 1 = 1 & a_7 = 1. \end{array}$$

Now let's carry out the algorithm on the number 89 again to find its base 5 equivalent. First divide 89 by 5 and record the remainder. Since $89 = 17 \cdot 5 + 4$, the remainder is 4. Then divide the quotient 17 by 5 and record that remainder. Since $17 = 3 \cdot 5 + 2$, we have the remainder 2. Finally divide 3 by 5 to get quotient 0 and remainder 3. Next record these numbers in reverse order 324_5 .

Next we consider the two methods for converting a fraction into another base. *Repeated subtraction* works much the same as it did in the case of whole numbers, but now the process may not always end so nicely. Let's consider the binary case as an example. Find the binary representation of $f = 23/32$. Since f is less than 1, we know the representation will start $0.b_1b_2b_3\dots$ where each b is either 0 or 1. As before, make a list of the powers of 2, except that this time, it's the negative integer powers of 2 that interest us. These numbers are $2^{-1} = 1/2 = 0.5$, $2^{-2} = 1/4 = 0.25$, $2^{-3} = 1/8 = 0.125$, etc. The first step in the process is to determine the largest of these fractions that does not exceed f . Since $f > 1/2$, the answer is the first one, $1/2$. Thus $f = 23/32 = 1/2 + (23/32 - 1/2) = 1/2 + (23/32 - 16/32) =$

$1/2 + 7/32$. Next find the largest fraction from the list above that is no bigger than $7/32$. Since $1/4 = 8/32$ is too big, we have the fraction $1/8$ to subtract. Note that $7/32 = 1/8 + (7/32 - 1/8) = 1/8 + (7/32 - 4/32) = 1/8 + 3/32$. Repeat this process the number $3/32$. The appropriate power of 2 is $1/16$, so we can write $3/32 = 1/16 + (3/32 - 1/16) = 1/16 + (3/32 - 2/32) = 1/16 + 1/32$. Finally we can write the original number as a sum of powers of 2:

$$23/32 = 1/2 + 1/8 + 1/16 + 1/32.$$

Note that the power 2^{-2} does not appear, so a place must be help for it. In other words,

$$23/32 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5}.$$

Finally, we can write the number f in binary notation: $f = 0.10111_2$. You're invited to try this method with other bases in the homework.

Next we consider the alternate method promised earlier, called repeated multiplication. Again we will consider the number $f = 23/32$. First, multiply by 2 and write down the integer part. Thus $23/32 \times 2 = 46/32 = 1 + 14/32$ so the integer part is 1 and the left-over fraction is $14/32$. Repeat the process, multiplying $14/32$ by 2 and peeling off the integer part: $14/32 \times 2 = 28/32 = 0 + 28/32$, so you see that the integer part is 0. The new fraction is $28/32$. Repeating the process, $28/32 \times 2 = 56/32 = 1 + 24/32$. Repeating, $24/32 \times 2 = 48/32 = 1 + 16/32$. The last step is $16/32 \times 2 = 1$. Now take these integer parts and line them up from left to right as $b_1b_2b_3 \dots$. This results in $f = 0.10111_2$ because the first integer part obtained was 1, the next was 0 and the final three were all 1's. Why does this method work? The answer is that multiplication by the base b has the effect of shifting the decimal (actually, its not called a decimal point unless we are talking base 10, but instead called a *radix point*). For example look at what multiplication by 10 does for the number $f = 0.56832$. The integer part of $10f$ is the number 5 which is the tenths digit of f . The fraction part of $10f$ is 0.6832, so after multiplying by 10 we obtain the integer part as 6, which is the hundredths digit. Continuing in this way, we can get each digit of f one at a time.

Operations in a binary system are essentially simpler than in the standard decimal one. If necessary to remember only the tables of addition and multiplication:

Table of Summation

+	0	1
0	0	1
1	1	10

Table of Multiplication

×	0	1
0	0	0
1	0	1

Numerical illustrations:

$$\begin{array}{r}
 \text{a) } \begin{array}{r} (1011)_2 \\ + (101)_2 \\ \hline (10000)_2 \end{array} \\
 \\
 \begin{array}{r} 1011 \\ \times 111 \\ \hline 1011 \\ 10110 \\ \hline 101100 \\ 1001101 \end{array}
 \end{array}
 \quad
 \begin{array}{l}
 \text{In decimal system} \\
 1011_2 = 2^3 + 2 + 1 = 11 \\
 101_2 = 2^2 + 1 = 5 \\
 11 + 5 = 16 = 2^4 = 10000_2 \\
 \\
 \text{In decimal system} \\
 1011_2 = 11, 111_2 = 7 \\
 11 \times 7 = 77, 77 = \\
 = 64 + 8 + 4 + 1 = 1001101_2
 \end{array}$$

Representations in other bases. If $1 < b \leq 10$ then we can use the standard notations for the digits, if $b > 10$ we have to introduce additional symbols. For instance, if $b = 12$ then together with the digits $0, 1, \dots, 9$ we'll use the two additional digits $\alpha = (10)_{10}$, $\beta = (11)_{10}$.

Numerical example: transition from $b = 12$ to $b = 10$:

$$\begin{aligned}
 n &= (\beta 50\alpha)_{12} = \beta \cdot 12^3 + 5 \cdot 12^2 + 0 \cdot 12 + \alpha = \\
 &= 11 \cdot 12^3 + 5 \cdot 12^2 + 10 = 19\,008 + 720 + 10 = \\
 &= (19738)_{10}
 \end{aligned}$$

For the important system base $b = 16$ (Hexadecimal system), we'll use the following notations for the digits:

$$0123456789 \overset{10}{A} \overset{11}{B} \overset{12}{C} \overset{13}{D} \overset{14}{E} \overset{15}{F},$$

or (in binary notation)

$$0000 \quad 0001 \quad 0010 \dots \dots 1110 \quad 1111.$$

Example 5.

$$\begin{aligned}
 n &= (7B5F)_{16} = 7 \cdot 16^3 + 11 \cdot 16^2 + 5 \cdot 16 + 15 = \\
 &= 28672 + 2816 + 80 + 15 = (31583)_{10}.
 \end{aligned}$$

The Second Algorithm (repeated division by b with remainder) gives the base- b of a decimal integer.

Example 6.

$$\text{a) } n = (1073)_{10}, b = 5$$

$$1073 = 5 \cdot 214 + 3$$

$$214 = 5 \cdot 42 + 4$$

$$42 = 5 \cdot 8 + 2$$

$$8 = 5 \cdot 1 + 3$$

$$1 = 1$$

$$1073 = (13243)_5 = 1 \cdot 5^4 + 3 \cdot 5^3 + 2 \cdot 5^2 + 4 \cdot 5 + 3$$

$$\begin{array}{r} 625 \\ \underline{375} \\ 1000 + 50 + 20 + 3 = 1073 \end{array}$$

b) $n = 1463$, $b = 12$

$$\begin{aligned} 1463 &= 12 \cdot 121 + 11 \\ 121 &= 12 \cdot 10 + 1 \\ 10 &= 10 \\ n &= (\alpha 1 \beta)_{12} \end{aligned}$$

c) $n = 4089$, $b = 16$

$$\begin{aligned} 4089 &= 16 \cdot 255 + 9 \\ 255 &= 16 \cdot 15 + 15 \\ 15 &= 15 \\ n &= (FF9)_{16} \end{aligned}$$

The multiplication and addition tables depend on b . The tables for $b = 5$ with digits 0, 1, 2, 3, 4 are given below.

Table of Addition ($b = 5$)

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	12
4	4	10	11	12	13

Table of Multiplication ($b = 5$)

·	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	11	13
3	0	3	11	14	22
4	0	4	13	22	31

Example 7. Transform $h = 67$, $m = 29$ to the base 5 and multiply in this base.

Solution:

$$\begin{aligned} 67 &= 2 \cdot 25 + 3 \cdot 5 + 2 = (232)_5 \\ 29 &= 1 \cdot 25 + 0 \cdot 5 + 4 = (104)_5 \end{aligned}$$

$$\begin{array}{r} 232 \\ \times 104 \\ \hline 2033 \\ 000 \\ 232 \\ \hline 30233_5 \end{array}$$

In this section we discuss a method for solving problems associated with divisibility and with remainders. Its called **Modular Arithmetic**. Let m be a positive integer, and let a and b be integers. We say ' a is congruent to b modulo m ' and

write $a \equiv b \pmod{m}$ if division of a and of b by m yield the same remainder. For example $55 \equiv 61 \pmod{3}$ because each of the numbers 55 and 61 yield a remainder of 1 when divided by 3. Here are some other examples. Be sure you see why they hold before reading on. $73 \equiv 73 \pmod{m}$; $6 \equiv -4 \pmod{5}$; $123123 \equiv 0 \pmod{11}$. Another way to test congruence is to note that $a \equiv b \pmod{m}$ is equivalent to saying that $m|(a-b)$. This equivalent condition is useful in proving the next two theorems.

Theorem 1 Let m be a positive integer. Then

1. Reflexivity. For all integers a , $a \equiv a \pmod{m}$.
2. Symmetry. For all integers a, b , if $a \equiv b \pmod{m}$, then $b \equiv a \pmod{m}$.
3. Transitivity. For any three integers, a, b , and c , if $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$, then $a \equiv c \pmod{m}$.

Proof. Part 1 follows from the fact that $m|(a-a)$ for any a and any positive m , which is true because $0/m = 0$ is an integer. Part 2 is true because $(a-b)/m = -(b-a)/m$ is an integer, and the negative of an integer is also an integer. To prove part 3, note that $m|a-b$ and $m|b-c$. The $m|(a-b)+(b-c)$. But $(a-b)+(b-c) = a-c$, so we are done. We'll see later in Lecture 9 that any relation satisfying all three of the properties, reflexivity, symmetry, and transitivity is very special. Such a relation is called an *equivalence relation*. Next we have a very practical theorem which we can apply immediately.

Theorem 2 Let m be a positive integer, and suppose $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$. Then

1. Additivity. $a + c \equiv b + d \pmod{m}$
2. Productivity. $ac \equiv bd \pmod{m}$; and in particular, $ac \equiv ad \pmod{m}$.
3. Exponentivity. $a^n \equiv b^n \pmod{m}$ holds for any positive integer n .

Proof. To see why 1. is true, note that $\frac{a+c-(b+d)}{m} = \frac{a-b}{m} + \frac{c-d}{m}$ is the sum of two integers, so is an integer itself. Part two follows similarly. Note that $ac - bd = ac - ad + ad - bd = a(c-d) + d(a-b)$ is the sum of two multiples of m and therefore is itself a multiple of m .

The first application of theorem 2 is the rule for divisibility by 9. A number N is divisible by 9 if and only if the sum of the decimal digits of N is divisible by 9. In fact we can say more. Let $S(N)$ denote the sum of the digits of N . Then $N \equiv S(N) \pmod{9}$. To see why this is true, let $N = a_n 10^n + a_{n-1} 10^{n-1} \cdots a_0$. Note that $1 \equiv 1 \pmod{9}$, $10 \equiv 1 \pmod{9}$, $10^2 \equiv 1 \pmod{9}$, \dots , and $10^n \equiv 1 \pmod{9}$. These congruences all follow from the productivity property or the exponent property above. Again from the productivity property, $a_0 \equiv a_0 \pmod{9}$, $10a_1 \equiv a_1 \pmod{9}$,

$10^2 a_2 \equiv a_2 \pmod{9}$, \dots , and $10^n a_n \equiv a_n \pmod{9}$. Finally, the additivity property finishes off the proposition, $N \equiv a_n + a_{n-1} + \dots + a_0 \pmod{9}$.

Try to mimic these ideas to prove a divisibility test for 11. Next we use what we've developed to work some remainder problems that involve exponentiation. For example, find the remainder when 6^{2003} is divided by 7. First note that $6 \equiv -1 \pmod{7}$. Then by the exponential part of theorem 2, we have $6^{2003} \equiv (-1)^{2003} \pmod{7}$. But $(-1)^{2003} = -1$, and $-1 \equiv 6 \pmod{7}$. Putting all this together gives us $6^{2003} \equiv 6 \pmod{7}$. This is another way to say that when 6^{2003} is divided by 7, the remainder is 6.

June 2, 2003 1:35 P.M.