# 4.5 Applications of congruences

Recall our notation from section 4.1:

- $a \mid b$ means $a$ divides $b$
- $a \bmod m$ means the remainder when you divide $a$ by $m$
- $a \equiv b \pmod m$ means $m$ divides $a-b$.

Note that $a \mid b$ if and only if $b \bmod a = 0$.

## Hashing functions

These have many applications in cryptography, data storage and error checking. The basic idea is to reduce bigger numbers to smaller ones of a fixed size. And a simple way to do that is using remainders.

Example ① Let the hashing function $h(k)$ be defined by $h(k) = k \bmod 100$. Find $h(123456)$ and $h(9172)$.

Solution: If you divide 123456 by 100 you get 1234.56 as a decimal so the quotient is 1234 and 56 is left over:

$$h(123456) = 56.$$

Also $h(9172) = 72$.

---

We know from the division algorithm that when you divide any integer by 100 the remainder must be between 0 and 99.

Clearly, $k \bmod 100$ just gives the two right most digits of $k$.

Of course $h$ in example ① is not one-to-one, with many numbers giving the same hashing values. For example

$$h(9172) = h(10072) = h(772)$$

and examples like this are called collisions.

- See examples 1, 2 in the book on p 288.

To find $64212848 \bmod 111$

using a calculator, do the division to get the decimal $578494.1261\ldots$

So the quotient is 578494 and the remainder is

$$64212848 - 111 \cdot 578494 = 14.$$

$$a - dq = r$$

## Pseudorandom numbers

Nothing that a computer does is random, but the remainder idea can be used to find sequences of integers that seem random and are good enough for applications.

In the linear congruential method we fix three positive integers

$m$ modulus
$a$ multiplier
$c$ increment

and use them to make a random looking sequence $x_0, x_1, x_2, \ldots$ like this.

Well we pick an $x_0$ called the seed.

Then $x_1 = (ax_0 + c) \bmod m$

so $0 \leq x_1 \leq m-1$

next $x_2 = (ax_1 + c) \bmod m$

$x_3 = (ax_2 + c) \bmod m$

$\vdots$

$$x_{n+1} = (ax_n + c) \bmod m$$

Each number in the sequence depends on the one before. This is called recursion.

Example ② Compute the first three numbers after the seed 11 using modulus 15, multiplier 13 and increment 5.

Solution: $x_0 = 11$

$$x_1 = (13x_0 + 5) \bmod 15$$
$$= 148 \bmod 15 = 13$$

$$x_2 = (13x_1 + 5) \bmod 15$$
$$= 174 \bmod 15 = 9$$

$$x_3 = (13 \cdot x_2 + 5) \bmod 15$$
$$= 122 \bmod 15 = 2$$

So our pseudorandom sequence starts
11, 13, 9, 2, ...

- see example 3 p 289.

## Check digits

Many identifying numbers, like bank accounts, credit card numbers, ISBN numbers for books, UPC numbers for retail products contain check digits. The check digits can usually detect if a number is not correct - say if you made a mistake typing in your VISA card number for an online purchase.

Check digits for credit card numbers use the Luhn algorithm. For example, this is a valid card number:

4298 1365 2981 0728

↑ check digit

To compute the check digit, first start with the 15 digits

4298 1365 2981 072 .

Double the underlined digits and if the result is more than 9 then add the two digits. So $8 \longrightarrow 16 \longrightarrow 1+6 = 7$.

we get 8298 2335 4971 074

now add these digits to get 72

and compute $72 \bmod 10 = 2$.

The check digit is chosen to make the digits sum $\bmod 10 = 0$:

$(72+8) \bmod 10 = 0$.

↑

If you make a mistake typing one digit of your card number then this sum mod 10 will not equal 0 and an error is detected. If you make many mistakes there is a 10% chance it won't be detected.

Example (3) Is this a valid credit card number?

6789 0123 4567 4321

Solution: Double the underlined digits and add the answer's digits if more than 9:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 4 | 3 | 2 | 1 |
| 12 | | 16 | | | | | | | | 12 | | | | | |
| 3 | 7 | 7 | 9 | 0 | 1 | 4 | 3 | 8 | 5 | 3 | 7 | 8 | 3 | 4 | 1 |

Add the digits $3+7+7+\cdots+3+4+1 = 73$

Now $73 \bmod 10 \neq 0$ so this is

not a valid credit card number.