The numbering of the world.

We've probably all been the subject of mistaken identity at one time or another. Our names are just not different enough, there aren't enough to go around?

John Smith (1 @ UNR)
David Smith (5!)

Germany!

- phone numbers
- SSNs

(heri.unl.edu
129.93.181.158
= = = =
AC we math nine.

numeric - all #s
alphanumeric - e.g., VIN#s.

nowadays nearly every transaction we have has a number attached to it.

examples?

Problem: wrong numbers!

phone/SSNs: the # of possible #s is close to the # in use! So an error ~~(typically: one # changed, transpose 2 #s)~~ often results in another valid (ie. in use) but wrong #.

~700 million SSNs possible
~320 million already in use!

Playing telephone....

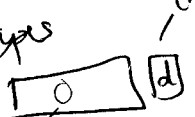
with an ID # the most common errors are
(1) a single digit is wrong
(2) two adjacent digits get switched (transposition error)

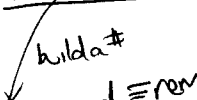
In many cases we just have to put up with such mistakes, but for many transactions this could be a major problem. - credit cards.

The solution is to try to design ID #s so that we can be reasonably sure that it hasn't been entered incorrectly, i.e. that (ideally) if $n_1 \cdots n_k$ is a valid number then changing one number or switching two adjacent ones isn't valid. Then the error can be detected.

Math to the rescue! A common technique is to make the ID # longer, by adding a check digit.

Two types





d = remainder of n on division by some fixed N

or

n+d = multiple of N

The idea:

original ID = 135268493

$1+3+5+2+6+8+4+9+3 = 41$

new ID# = 1352684931

last digit of ID = last digit of sum of old digits

now if any one digit is changed, we can notice this.

1357684931

sum = 46 $\quad 6 \neq 1$ so wrong.

Cleverly-designed checkdigits can detect not only a single change but also a transposition error.

And these schemes are all around you!

ISBN # = 10 digits (well, 9 digits + (0-9, X))

$a_1 \cdots a_9 a_{10}$ a_{10} = check digit, chosen so that

$10 \cdot a_1 + 9 \cdot a_2 + \cdots + 3 \cdot a_8 + 2 \cdot a_9 + a_{10}$ = multiple of 11

(catches all single digit & transp errors) ($X = 10$)

(since 1/1/07: need more numbers!)

ISBN = $a_1 \cdots a_{12} a_{13}$

$1 \cdot a_1 + 3 \cdot a_2 + 1 \cdot a_3 + \cdots + 1 \cdot a_{11} + 3 \cdot a_{12} + a_{13}$ = multiple of 10.

(catches all single digit and most transp. errors)

UPC's: 12 digit code

$a_1 - a_2 \cdots a_6 - a_7 \cdots a_{11} - a_{12}$

product category | mfr | item | check digit

a_{12} chosen so that

$3(a_1 + a_3 + a_5 + \cdots + a_{11}) + 1 \cdot (a_2 + a_4 + \cdots + a_{12})$ is a multiple of 10.

need

LUHN code (many credit cards)

$a_1 \cdots a_{16}$

a_{16} = check digit so that

$a_{16} + b_1 + \cdots + b_{15}$ is a multiple of 10,

where

$b_1 = 2a_1$ if $2a_1 < 10$, o/w b_1 = sum of the 2 digits

$b_2 = 2a_2$ if $2a_2 < 10$, o/w b_2 = sum of the 2 digits

$b_3 = 2a_3$ if

$b_{15} = 2a_{15}$ $2a_{15} < 10$

e.g., for add 1 1 2 9 3 3 5 1 4 2 7 3 5 1 2 = 49; to get mult of 10, add 1

10 2 12 14 4 16 14 2

5129 6371 2283 711?

we find $2 \cdot 5 = 10$ so $b_1 = 1 + 0 = 1$

$2 \cdot 1 = 2$ so $b_2 = 2$

4	4
18	9
12	3
6	6
14	5
2	2
4	4
4	4
16	7
6	6

14
2
2
61

so to get a mult of 10, add 9

check digit is 9.

any one number can be determined from the others: (ex!)

⇒ can detect single digit errors

designed We

HW: 1.1: 4, 5, 9, ~~[illegible]~~, 22, 23*, 27, 33, 36

16 17* typo in 24*

turn in.

description of ISBN before #15 is wrong!

2.2. Why this stuff works: modular arithmetic.

Its all based on

The division algorithm: if a & m are whole numbers with $m \neq 0$, then there are unique whole numbers q (=quotient) and r (=remainder)

∆ so that $a = qm + r$ (i.e. $\frac{a}{m} = q + \frac{r}{m}$)

with $0 \leq r \leq m-1$

whole number part — fractional part.

The key to the check digit schemes is the unique part.

In general all we will care about is the remainder, so one way in practice to find it is, starting with a, keep subtracting m, or multiples of m, until you are left with something btw 0 and $m-1$.

Check digits:

$a_1 \cdots a_8 a_9$ ← check: $a_1 + a_2 + \cdots + a_8 + a_9 =$ mult of 10.

detects change in one digit (sum won't be a mult of 10)

catch transposition errors? need to treat adjacent #s differently.

ISBN $a_1 \cdots a_{10}$ ← check (0–9, X) "10" choose a_{10} so that

$10a_1 + 9a_2 + \cdots + 2a_9 + a_{10} =$ mult of 11.

UPC $a_1 a_2 \cdots a_{12}$ ← check choose a_{12} so that

$3a_1 + a_2 + 3a_3 + a_4 + 3a_5 + \cdots + 3a_{11} + a_{12} =$ mult of 10.

And

Is this good enough? Have we lost the ability to catch a single change? If so/not, why/why not?

To help: modular arithmetic.

Starting point: division, elementary school style.

$7 \div 3 = 2$ with remainder 1 (not 2.3333...!) fractional pt

Div. Alg.

Given whole numbers a and m with $m \neq 0$ there are unique whole numbers q (= quotient) and r (= remainder), whole #s with $0 \leq r \leq m-1$ and $a = q \cdot m + r$. $\left(\text{i.e. } \frac{a}{m} = q + \frac{r}{m}\right)$

we say $r = a \bmod m$; it's what a "leaves behind" when you ~~divide by m~~ take away mults of m from a.

uniqueness! $q_1 m + r_1 = a = q_2 m + r_2$ $\quad 0 \le r_1, r_2 \le m-1$ then

$$r_1 - r_2 = m(q_1 - q_2) \quad \text{but}$$

$$\underbrace{0-(m-1)}_{-(m-1)} \le r_1 - r_2 \le \underbrace{(m-1)-0}_{m-1}$$

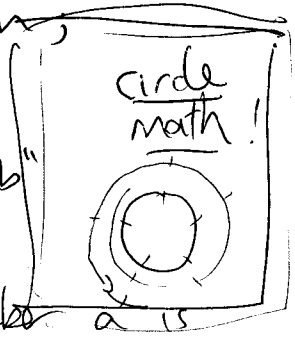
so $r_1 - r_2$ is a multiple of m b/w $-(m-1)$ and $m-1$. The only one of these is 0. So $0 = m(q_1 - q_2)$ & $m \neq 0$.

so $q_1 - q_2 = 0$.

Note that if $r = a \bmod m$ then $a - r$ is a mult of m.

More generally, we say that a and b are congruent modulo (or mod) m if $a-b$ is a multiple of m;

$a \equiv b \pmod m$ means $m \mid a-b$ ("m divides $a-b$")



To decide what # b/w 0 and $m-1$ some number is congruent to, we can ~~sep~~ divide $\frac{a}{m}$, throw away whole # part and mult fractional pt by m again to recover r. (if a is negative!) or you can just continually subtract/add multiples of m from a until you're left with something b/w 0 & $m-1$.

eg. mod 13, $2357 \equiv 2357 - 1300 = 1057 \equiv 1057 - 13\cdot 80$
$= 1057 - 1040 \equiv 17 \equiv 17 - 13 = 4$.

Our check digit schemes can be described in terms of congruence; since "is a multiple of blah" is the same as "is $\equiv 0 \pmod{\text{blah}}$"

& eg. ISBN-10: $a_1 \cdots a_{10}$ must satisfy
check

$$10a_1 + 9a_2 + \cdots + 2a_9 + a_{10} \equiv 0 \pmod{11}$$

A few basic facts about $\equiv$ allow us to see that ISBN-10 does what we want!

~~Suppose we change swap two numbers~~

If $a \equiv b \pmod m$ and $b \equiv c \pmod m$ then $a \equiv c \pmod m$

If $a_1 \equiv a_2 \bmod m$ and $b_1 \equiv b_2 \bmod m$,

then $a_1 + b_1 \equiv a_2 + b_2 \bmod m$

$a_1 - b_1 \equiv a_2 - b_2 \bmod m$

$a_1 b_1 \equiv a_2 b_2 \bmod m$

$m \mid a_1 - a_2$

$m \mid b_1 - b_2$

$m \mid (a_1 + b_1) - (a_2 + b_2)$

$(a_1 - a_2) + (b_1 - b_2)$

e.g. $m = 7$ $\quad 23 \equiv 2$

$31 \equiv 3$

$23 \cdot 31 - 2 \cdot 3 = (23-2)\cdot 31 + 2\cdot(31-3)$

mult of 7 $\qquad$ mult of 7

Now we can see that ISBN-10 does what we want!

Suppose we ~~change~~ swap two numbers

~~$10a_1+9a_2+8a_3$~~

$10a_1+9a_2+8a_3+7a_4+6a_5+5a_6+4a_7+3a_8+2a_9+a_{10}\equiv 0$

$-(\ 10a_1+9a_2+8a_3+7a_5+6a_4+5a_6+4a_7+\cdots+a_{10}\equiv 0)$

$\equiv 0-0$

$=0$

a_4-a_5

So a_4-a_5 is a mult of 11

but $-9 \le a_4-a_5 \le 9$ $\Rightarrow$ $a_4-a_5=0$

$a_4=a_5$

Change a single #? Some sort of idea!

$10a_1+9a_2+\cdots+5a_6+\cdots+a_{10}\equiv 0$

$-(\ 10a_1+9a_2+\cdots+5a_6'+\cdots+a_{10}\equiv 0\ )$

$5(a_6-a_6')\equiv 0$

So $5(a_6-a_6')$ is a mult of 11.

But since 11 is prime, $11 \mid 5(a_6-a_6')$ and $1\le 5\le 10$ means that $11 \mid a_6-a_6'$ so, as before, $a_6=a_6'$.

Or: $5(a_6-a_6')\equiv 0$ then $8\cdot 5(a_6-a_6') \equiv 8\cdot 0 = 0$, so $a_6-a_6\equiv 0$

$40(a_6-a_6') \equiv 1\cdot(a_6-a_6')$

So ISBN-10 can detect both kinds of error.

$\Rightarrow$

Similarly, UPC $a_1 \cdots a_{12}$

$$3a_1 + a_2 + 3a_3 + \cdots + 3a_{11} + a_{12} \equiv 0 \pmod{10}$$

detects all changes in single digits

$$a_6 - a_6' \equiv 0 \rightsquigarrow a_6 - a_6'$$
$$3a_7 - 3a_7' \equiv 0 \rightsquigarrow 7 \cdot (3a_7 - 3a_7') \equiv 0$$
$$21(a_7 - a_7') \equiv (a_7 - a_7')$$

most, but not all

~~if it~~ detects all transposition errors:

$$\begin{array}{l} +3a_5 + a_6 + \\ +3a_6 + a_5 + \\ \hline 2(a_5 - a_6) \end{array} \equiv 0 \mod 10$$

Mod-9 check digit scheme

$a_1 \cdots a_n$ = document # | check digit (0-8)

so that the entire number is $\equiv 0 \pmod 9$

This is actually quite similar to the first scheme we introduced, since a number is $\equiv \pmod 9$ to the sum of its digits:

$$36514_2 = 36514 \cdot 10 + 2 \equiv 36514 \cdot 1 + 2 \pmod 9$$

(since $10 \equiv 1 \bmod 9$) and keep going.

Ex US Postal Money orders use an 11-digit mod 9 scheme.

there's nothing really special about 9, others eg. UPS tracking #s use a mod 7 check digit scheme

$a_1 \ldots a_n \equiv 0 \pmod 7$

a_n: check

The mod 9 check almost detects changing a digit

$a_1 + a_2 + \cdots + a_{n-1} + a_n \equiv 0 \pmod 9$

changing changes the $\equiv 0 \pmod 9$, unless one is 0 and other is 9

doesn't detect swaps.

9
18
27
36 ≡ 1

mod 7 check detects fewer digit changes

$a_1 \cdots 2 \cdots a_n \equiv 0 \pmod 7$

then

$a_1 \cdots 9 \cdots a_n \equiv 0 \pmod 7$

(they differ by $70\cdots0$)

Better at swaps $a_1 \cdots 27 \cdots a_n \equiv 0$

$\Rightarrow a_1 \cdots 72 \cdots a_n \equiv 0$?

difference is $450\cdots0$, a mult of 7.

$a_1 a_2 \equiv a_2 a_1 \pmod 7$?

$10a_1 + a_2 \equiv 10a_2 + a_1 \pmod 7$

$9a_1 \equiv 9a_2$

$36a_1 \equiv 36a_2$ $a_1 \equiv a_2$ ✓

(if $10x \equiv 10y \pmod 7$ then $x \equiv y \pmod 7$,

since $5 \cdot 10x \equiv 5 \cdot 10y = 50y \equiv 1 \cdot y = y$ since $50 \equiv 1 \pmod 7$

$50x \equiv 1 \cdot x = y$

1.2 #18

$39 \cdot 73 \equiv b \pmod 8$

$39 = 32 + 7 \equiv 7 \pmod 8$ $\quad 73 = 72 + 1 \equiv 1 \pmod 8$

$39 \cdot 73 \equiv 7 \cdot 1 = 7 \pmod 8$ $\quad \underline{b = 7}$

$29 \cdot c \equiv 2 \pmod 5$

$29 \cdot 0 = 0 \equiv 0$
$29 \cdot 1 = 29 = 25 + 4 \equiv 4$
$29 \cdot 2 = 58 = 55 + 3 \equiv 3$
$29 \cdot 3 = 87 = 85 + 2 \equiv 2 \leftarrow c = 3$
$29 \cdot 4 = 116 = 115 + 1 \equiv 1$

$29 = 25 + 4 \equiv 4 \pmod 5$

$4c \equiv 2 \pmod 5$

$4 \cdot 4c \equiv 4 \cdot 2 = 8 \equiv 3$

$16c$ $\quad c = 3$

$1 \cdot c = c$

HW 1.2 p31+: 3, 5, 6, 10, 11, 17, 18*,

assigned

mod 10 check: 21, 27, 30, 32, 33, 36, 37, 38*

~~Barcodes:~~

mod 9 euros
visa travelers checks

mod 7 ups tracking #s
airline tickets

~~For Fri: 1.2 p31+: 21, 27, 30,~~