Math 189H Joy of Numbers Activity Log

Thursday, November 3, 2011

David Hilbert: "The art of doing mathematics consists in finding that special case which contains all the germs of generality."

Plato: "I have hardly ever known a mathematician who was capable of reasoning." [It should be noted, though, that in Plato's time, astrologers were (also) called mathematicians...]

Goldbach's Conjecture: Every even number is the sum of two primes. Verified by hand in 1938 to 10,000. Verified by computer (in 2010) to $2 \cdot 10^{18}$. Still unsolved!

Today we took a slight break from our steady march towards our goal of building the machinery needed to understand how the things we are learning impact electronic security to look at what we called "the number theory in your pocket". The fact is that we, and pretty much everything around us, are all numbers, at least as far as many of the organizations we interact with are concerned. To the government we 'are' a social security number, to businesses we 'are' a credit card number, to the grocery store the things we purchase 'are' UPC (which stands for 'universal product code') codes [which makes saying 'UPC code' redundant...], our textbook 'is' an ISBN (='international standard book number') number [same problem...!]. These numbers are constantly being scanned and typed into order forms; so how do we make sure that they are being entered correctly? The fact is that most of these numbers (with the notable exception of your SSN!) have a 'check' digit built into them, usually the last digit of the number, which is designed to protect us from the kinds of common errors that people (and machines) make when entering data; the last digit is designed to detect (and therefore reject) erroneously entered data. We made ourselves a list of the kinds of errors we might expect to encounter when transmitting a number to someone else (which I augmented for our log from the Wikipedia entry for 'check digit'):

dropping a digit, such as 1234 changed to 134
single digit errors, such as 1 changed to 2
    (these are often based on 'shape': $3 \leftrightarrow 8$, $1 \leftrightarrow 7$, $2 \leftrightarrow 5$, $6 \leftrightarrow 9$)
transposition errors, such as 12 changed to 21
twin errors, such as 11 changed to 22
jump transposition errors, such as 132 changed to 231
jump twin errors, such as 131 changed to 232
phonetic errors, such as 60 changed to 16 ("sixty" to "sixteen")
inserting a digit, such as 1234 changed to 15234
doubling, such as 123 changed to 1223
'compression', such as 1223 changed to 123

Of these probably the most prevalent are the single digit and transposition errors, and most check digits do a good job of detecting such errors. This all lies in the method used to compute the check digit, which in nearly all implementations use a modular weighted sum. That is, for a string of digits $a_1a_2 \cdots a_n$, a check digit a_{n+1} is added to the end that is computed by summing specific multiples $x_1a_1 + x_2a_2 + \cdots + x_na_n$ of the digits, modulo a specific modulus N; typically the check digit a_{n+1} is chosen so that N divides $x_1a_1 + x_2a_2 + \cdots + x_na_n + a_{n+1}$.

The design principles involved in choosing weights and modulus can be inderstood by studying a few examples. For credit cards, which have 16 digits, the method most commonly used is called the *Luhn algorithm*, which uses a modulus of $N = 10$ and weights $2, 1, 2, 1, \ldots, 2, [1]$ (to tidy things up, we usually think in terms of including the check digit, with a weight of 1, and think of $x_1a_1 + x_2a_2 + \cdots + x_na_n + a_{n+1} \bmod N$ as a 'test' that the number (with check digit) is 'valid'; it is valild if the sum is 0 modulo N). In other words, the credit card number $a_1a_2 \cdots a_{16}$ is valid if

$2a_1 + a_2 + 2a_3 + a_4 + \cdots + 2a_{15} + a_{16} = 2(a_1 + a_3 + \cdots + a_{15}) + (a_2 + a_4 + \cdots + a_16)$ is a multiple of 10.

[Several of you reported, though, that this was not true of your own credit card numbers?!] A slightly different way to think of this is that a_{16} is chosen so that, when added to the last digit of $2a_1 + a_2 + 2a_3 + a_4 + \cdots + 2a_{15}$, you get 10.

Why does this work to detect common errors? Basically, it detects transposition errors because it treats adjacent digits differently; if we replace a_ia_{i+1} with $a_{i+1}a_i$, our 'check sum' $2a_1 + a_2 + 2a_3 + a_4 + \cdots + 2a_{15} + a_{16}$ will change by (plus or minus) $a_i - a_{i+1}$ (if we subtract the new sum from the old one), and so, assuming the original number was valid, so its check sum was a multiple of 10, the only way this new number passes the same test is if $a_i - a_{i+1}$ is a multiple of 10. But since both digits are between 0 and 9, the only way this will be true is if $a_i = a_{i+1}$, which means that transposing the digits didn't change the number! Similar reasoning enables us to understand the effect of a single digit error: if a_i is replaced by a_i', then the check sum will change by (plus or minus) either $a_i - a_i'$ or $2(a_i - a_i')$, depending on the positiob of the changed digit. In the first case, this error will always be detected, since the new sum will not be a multiple of 10; in the second case, there is one error (out of the 9 possible other digits we could have chosen) that cannot be detected, namely if $a_i - a_i' = \pm 5$. All other single digit errors will be detected. So the Luhn algorithm detects all (adjacent) transposition errors, and 17/18-ths of all single digit errors.

The key design elements of this are that adjacent numbers are given different weights, and the weights mostly don't interfere with detecting a single digit error, in the sense that $N|w(a - b)$ usually allows us to conclude that $N|a - b$, which in turn implies that $a = b$, because the numbers a and b are fairly small. We have, in fact, seen how to insure that $N|w(a - b)$ implies $N|a - b$, by knowing that $\gcd(w, n) = 1$; the next check digit scheme is designed to use this to its ultimate.

An ISBN-10 number is a 10-digit number $a_1 \cdots a_{10}$ with the 10-th digit a check digit. The check sum uses weights $(10, 9, 8, \ldots, 2, 1)$ and modulus 11, and so an ISBN number is valid if $10a_1 + 9a_2 + \cdots + 2a_9 + a_{10} \underset{11}{\equiv} 0$. [We tested this with the instructor's copy of *The Penguin Dictionary of Curious and Interesting Numbers*.] This causes one slight problem, though; since $10a_1 + 9a_2 + \cdots + 2a_9$ can be congruent, mod 11, to 0 through 10, the check digit a_{10} must be able to take on the values 0 and 10 through 1 to make the sum a multiple of 11. But 10 isn't a 'digit'! The convention chosen to deal with this is that if $10a_1 + 9a_2 + \cdots + 2a_9 \underset{11}{\equiv} 1$, so that the check digit 'should' be 10, the check digit is denoted 'X'. [One of you, it turned out, had a book along that had check digit 'X'!] The additional overhead this causes is more than offset by the error-detecting abilities of this method; since 11 is prime, it is relatively prime to every weight being used in this check sum, and, even more, is relative prime to the difference of any two of the weights. This means that the check sum can detect every single digit error, since $11|w(a-b)$ and $1 \leq w \leq 10$ implies that $11|a-b$. It can also detect every trasposition error, even if the digits are not adjacent to one another, since the difference between the two check sums will be (plus or minus) $(w_i - w_j)(a_j - a_i)$, and so the only way that both check sums will be divisible by 11 is if $11|(w_i - w_j)(a_j - a_i)$; since $|w_i - w_j| \leq 10$, this means $11|a_j - a_i$, so $a_i = a_j$. So the ISBN check digit can detect all jump transposition errors, as well.

A final check digit scheme we dissected was the check digit used in UPCs. UPCs can have varying lengths, but a typical code has 12 digits, with the last digit the check digit. The first digit represents the basic type of product, the next five are a code for the manufacturer, and the next five after that represent the manufacturer's products, followed by the check digit. It uses a weighted sum scheme, with weights $(3, 1, 3, 1 \ldots, 3, 1)$ and a modulus of 10. That is, a UPC-12 code $a_1 a_2 \cdots a_{12}$ is valid if $3a_1 + a_2 + 3a_3 + \cdots + 3a_{11} + a_{12} \equiv 0 \bmod 10$. [We tested this on a box of pencils.] This code, because all of the weights are relatively prime to 10, can detect all single digit errors; but because adjacent weights differ by 2, which is not relatively prime to 10, it cannot detect all transposition errors. And because every other weight is the same, it cannot detect jump transposition errors at all. This trade-off - better detection of single digit errors, worse for transposition errors (compared to the Luhn algorithm) - is (probably!) a choice made due to the fact that UPC codes are scanned, so the machine is unlikely to transpose digits; catching single digit errors is more important when transposition errors are less likely!

In 2007, ISBNs moved to a 13-digit ISBN-13 number, with a check digit computed just as with UPCs, using weights $(1, 3, 1, 3, \ldots, 3, 1)$ and a modulus of 10. (Basically, they really became UPCs; what information the earlier digits 'encode' is just slightly different.)

As we originally remarked, US social security numbers do not include a check digit. There are some interesting restrictions on SSNs, though: no SSN starts with 666, or 000, and in fact xxx-00-xxxx and xxx-xx-0000 are also invalid SSNs. It used to be that the first three digits of your SSN described the location where your number was issued to you, but effective July, 2011, this is no longer the case; new SSNs will be issued randomly. Many other countries do incorporate a check digit into the personal/taxpayer identification numbers that they issue to their citizens. Australia, for example, employs a check digit computed using a weighted sum, but the weights used are officially a secret! (This is despite the fact that they have shared the secret with some 20,000 employers throughout the country, so that they can verify their employee's ID numbers...)

The point, really, is that modular arithmetic is taking place around you all the time! And the fact that $n|ab$ and $\gcd(a, n) = 1$ implies $n|b$ is a central fact underlying the effectiveness of all of these check digit schemes. Little did Euler and his colleagues know that they were laying the groundwork for modern commerce?