Luhn's Algorithm

Carmen Bruni

Real World Examples of Number Theory

- Question: With such an abstract concept like number theory, what can one do with it in the real world?

Real World Examples of Number Theory

- Question: With such an abstract concept like number theory, what can one do with it in the real world?
- Cryptographic applications are very common uses of number theory (so how data is securely transferred via the internet).

Real World Examples of Number Theory

- Question: With such an abstract concept like number theory, what can one do with it in the real world?
- Cryptographic applications are very common uses of number theory (so how data is securely transferred via the internet).
- There are however some other lesser known applications that we use every day.

- Let's take a look in your purse or wallet - what do you have inside? Some money? Pictures perhaps? Photo ID. Credit card...

Real World Examples of Number Theory

- Let's take a look in your purse or wallet - what do you have inside? Some money? Pictures perhaps? Photo ID. Credit card...
- That little piece of plastic that you use to pay for purchases actually has a ton of mathematics hidden in its implementation. Let's look at the number of my credit card.

My Visa number - shhhh!

- My Visa card number is the 16 digit number 4012888888881881.

My Visa number - shhhh!

- My Visa card number is the 16 digit number 4012888888881881.
- Let's flip the number around to get 1881888888882104.

My Visa number - shhhh!

- My Visa card number is the 16 digit number 4012888888881881.
- Let's flip the number around to get 1881888888882104.
- Take all the odd placed digits and add them up $1 + 8 + 8 + 8 + 8 + 8 + 2 + 0 = 43$.

My Visa number - shhhh!

- My Visa card number is the 16 digit number 4012888888881881.
- Let's flip the number around to get 1881888888882104.
- Take all the odd placed digits and add them up $1 + 8 + 8 + 8 + 8 + 8 + 2 + 0 = 43$.
- For all the even placed digits, double them, then add their digits $(1+6)+2+(1+6)+(1+6)+(1+6)+(1+6)+2+8 = 47$.

My Visa number - shhhh!

- My Visa card number is the 16 digit number 4012888888881881.
- Let's flip the number around to get 1881888888882104.
- Take all the odd placed digits and add them up $1 + 8 + 8 + 8 + 8 + 8 + 2 + 0 = 43$.
- For all the even placed digits, double them, then add their digits $(1 + 6) + 2 + (1 + 6) + (1 + 6) + (1 + 6) + (1 + 6) + 2 + 8 = 47$.
- Add these two numbers up to get 90, a number divisible by 10!
- Was this a coincidence? No! Try it on your own credit card (but be sure to rip the paper up afterwards!)

- The procedure outlined above is called Luhn's algorithm.

Luhn's Algorithm

- The procedure outlined above is called Luhn's algorithm.
- It is used so that it can detect simple typing errors made by the user.

Luhn's Algorithm

- The procedure outlined above is called Luhn's algorithm.
- It is used so that it can detect simple typing errors made by the user.
- A computer can easily verify (without a call to a database) whether a user has made a mistake on exactly one number or swapped two adjacent numbers (except for swapping a 9 and 0). These two mistakes are the most common types of entry errors.

Luhn's Algorithm

- The procedure outlined above is called Luhn's algorithm.
- It is used so that it can detect simple typing errors made by the user.
- A computer can easily verify (without a call to a database) whether a user has made a mistake on exactly one number or swapped two adjacent numbers (except for swapping a 9 and 0). These two mistakes are the most common types of entry errors.
- The last digit in a credit card number is a check sum and is chosen so that this algorithm works.

How it works

- First, for items in the even positions (after swapping), notice that

a_i	0	1	2	3	4	5	6	7	8	9
$h(a_i)$	0	2	4	6	8	1	3	5	7	9

  where here we denote by a_i the digit and by $h(a_i)$ by the value obtained by taking the digital sum of twice the number a_i.

How it works

- First, for items in the even positions (after swapping), notice that

a_i	0	1	2	3	4	5	6	7	8	9
$h(a_i)$	0	2	4	6	8	1	3	5	7	9

  where here we denote by a_i the digit and by $h(a_i)$ by the value obtained by taking the digital sum of twice the number a_i.
- In particular, notice that 0 and 9 remain unchanged but all other digits change position.

How it works

- First, for items in the even positions (after swapping), notice that

a_i	0	1	2	3	4	5	6	7	8	9
$h(a_i)$	0	2	4	6	8	1	3	5	7	9

  where here we denote by a_i the digit and by $h(a_i)$ by the value obtained by taking the digital sum of twice the number a_i.
- In particular, notice that 0 and 9 remain unchanged but all other digits change position.
- Thus, the sum total of the algorithm is given by

$$H = a_1 + a_3 + \ldots + a_{15} + h(a_2) + h(a_4) + \ldots + h(a_{16})$$

  or if you know fancy sigma notation,

$$H = \sum_{i=1}^{8} (a_{2i-1} + h(a_{2i}))$$

How it works

- First, for items in the even positions (after swapping), notice that

a_i	0	1	2	3	4	5	6	7	8	9
$h(a_i)$	0	2	4	6	8	1	3	5	7	9

  where here we denote by a_i the digit and by $h(a_i)$ by the value obtained by taking the digital sum of twice the number a_i.
- In particular, notice that 0 and 9 remain unchanged but all other digits change position.
- Thus, the sum total of the algorithm is given by

$$H = a_1 + a_3 + ... + a_{15} + h(a_2) + h(a_4) + ... + h(a_{16})$$

  or if you know fancy sigma notation,

$$H = \sum_{i=1}^{8} (a_{2i-1} + h(a_{2i}))$$

- Here we use the letter H to represented the 'hashed value'.

Changing a digit

- Let's examine each type of error and see how it changes H.
- Suppose we change a digit, say a_i is replaced by b_i distinct from a_i. Let's call H' the new hash number. Then either

$$H' = H - a_i + b_i \qquad \text{or} \qquad H' = H - h(a_i) + h(b_i)$$

- If we look at only the remainder when we divide by 10, since H is divisible by 10, we know that the remainder of H' when divided by 10 is the same as the remainder of when $-a_i + b_i$ (or in the other case $-h(a_i) + h(b_i)$). In either case though, these two values are different and thus the difference is not divisible by 10.
- Thus the new hash H' fails the check sum procedure and the algorithm flags the number as invalid.

Changing a digit

- Let's examine each type of error and see how it changes H.
- Suppose we change a digit, say a_i is replaced by b_i distinct from a_i. Let's call H' the new hash number. Then either

$$H' = H - a_i + b_i \qquad \text{or} \qquad H' = H - h(a_i) + h(b_i)$$

- If we look at only the remainder when we divide by 10, since H is divisible by 10, we know that the remainder of H' when divided by 10 is the same as the remainder of when $-a_i + b_i$ (or in the other case $-h(a_i) + h(b_i)$). In either case though, these two values are different and thus the difference is not divisible by 10.

Changing a digit

- Let's examine each type of error and see how it changes H.
- Suppose we change a digit, say a_i is replaced by b_i distinct from a_i. Let's call H' the new hash number. Then either

$$H' = H - a_i + b_i \qquad \text{or} \qquad H' = H - h(a_i) + h(b_i)$$

- If we look at only the remainder when we divide by 10, since H is divisible by 10, we know that the remainder of H' when divided by 10 is the same as the remainder of when $-a_i + b_i$ (or in the other case $-h(a_i) + h(b_i)$). In either case though, these two values are different and thus the difference is not divisible by 10.
- Thus the new hash H' fails the check sum procedure and the algorithm flags the number as invalid.

Swapping adjacent digits

- Suppose we swap adjacent digits, say a_i and a_{i+1} are swapped (and of course, assume these are distinct). Let's once again call H' the new hash number. We'll assume that i here is odd (the process is nearly identical if i is even). Then

$$H' = H - a_i - h(a_{i+1}) + h(a_i) + a_{i+1}$$

Swapping adjacent digits

- Suppose we swap adjacent digits, say a_i and a_{i+1} are swapped (and of course, assume these are distinct). Let's once again call H' the new hash number. We'll assume that i here is odd (the process is nearly identical if i is even). Then

$$H' = H - a_i - h(a_{i+1}) + h(a_i) + a_{i+1}$$

- If we look at only the remainder when we divide by 10, since H is divisible by 10, we know that the remainder of H' when divided by 10 is the same as the remainder of when $-a_i - h(a_{i+1}) + h(a_i) + a_{i+1}$.

Swapping adjacent digits

- Suppose we swap adjacent digits, say a_i and a_{i+1} are swapped (and of course, assume these are distinct). Let's once again call H' the new hash number. We'll assume that i here is odd (the process is nearly identical if i is even). Then

$$H' = H - a_i - h(a_{i+1}) + h(a_i) + a_{i+1}$$

- If we look at only the remainder when we divide by 10, since H is divisible by 10, we know that the remainder of H' when divided by 10 is the same as the remainder of when $-a_i - h(a_{i+1}) + h(a_i) + a_{i+1}$.
- Checking all such combinations of a_i and a_{i+1} reveals that unless these numbers are the same or they are 0 and 9, the new hash H' fails the check sum procedure and thus the algorithm flags the number as invalid.

Other uses

This algorithm is also used in many other forms of applications including

- Other major credit cards like American Express, Discover, Mastercard, etc.

Other uses

This algorithm is also used in many other forms of applications including

- Other major credit cards like American Express, Discover, Mastercard, etc.
- Your SIN number

Other uses

This algorithm is also used in many other forms of applications including

- Other major credit cards like American Express, Discover, Mastercard, etc.
- Your SIN number
- Debit cards

Other uses

This algorithm is also used in many other forms of applications including

- Other major credit cards like American Express, Discover, Mastercard, etc.
- Your SIN number
- Debit cards
- Types of personal identification numbers