Smart Codes

Adapted from a handout by Yingkun Li
Olga Radko Math Circle Advanced 2

April 25, 2021

1 Error Detection

Sequences of numbers, or *codes*, are ubiquitous in our lives. Some of them, such as credit card number and bank account numbers, are very important. They are also very sensitive to man-made and mechanical errors. So it is important that some of these numbers have the ability to detect, and even correct possible errors.

A nice example of codes with the ability to detect errors is the ISBN, a number associated to every book. There are two versions: one with 10 digits, one with 13 digits. For example, the 10-digit ISBN the first *Twilight* novel is 0316160172. The first digit 0 tells us that the book is in English. The next eight digits contains the publishing information of the book, and the last digit is used to to check that this is a *valid* ISBN. Suppose the first 9 numbers in a sequence of 10 digits looks like

$$\overline{a_1 a_2 \dots a_9}.$$

Calculate the following sum modulo 11

$$\sum_{k=1}^{9} k a_k. \tag{1}$$

The sequence is a valid ISBN if and only if the remainder agrees with the last digit, where we use the digit X to represent a remainder of 10.

In the *Twilight* example above, we have

$$\overline{a_1 a_2 \dots a_9} = 031616017. \tag{2}$$

The number in the expression (1) is

$$\begin{aligned} \sum_{k=1}^{9} k a_k &= 1 \cdot 0 + 2 \cdot 3 + 3 \cdot 1 + 4 \cdot 6 + 5 \cdot 1 + 6 \cdot 6 + 7 \cdot 0 + 8 \cdot 1 + 9 \cdot 7 && (3) \\ &= 145 && (4) \\ &\equiv 2 \pmod{11}. && (5) \end{aligned}$$

Since the remainder agrees the last digit 2, this is a valid ISBN.

Problem 1. Calculate the sum (1) for the following sequences of numbers to see if it is a valid ISBN.

(a) 0439023521

(b) 311001436X

(c) 0439784542

Problem 2. Given a sequence $\overline{a_1a_2\dots a_{10}}$, consider the sum

$$10 \cdot a_1 + 9 \cdot a_2 + \cdots + 2 \cdot a_9 + a_{10}. \tag{6}$$

What is its remainder when divided by 11 if $\overline{a_1a_2\dots a_{10}}$ is the number in Problem 1(a), 1(b), and 1(c) respectively?

Problem 3. (a) If the sequence $\overline{a_1a_2\dots a_{10}}$ is a valid ISBN, show that the sum in (6) is divisible by 11.

(b) If the sum in (6) is divisible by 11, is the ISBN necessarily valid?

Problem 4. (a) Is it possible to change a single digit in a valid ISBN such that it is still valid?

(b) Is it possible to make an invalid ISBN into a valid ISBN?

(c) What happens to a valid ISBN if you swap two adjacent digits? This is called a *transposition error*.

Besides the 10-digit ISBN, the 13-digit ISBN is also in use. For example, the 13-digit ISBN of the fourth Harry Potter book is 9780439139609, compare to the 10-digit ISBN 0439139600. The first three digits of the 13-digit ISBN provide more information about the book, and the last digit is still used for an error-detection. However, its error-detection mechanism is different from the one we described above. Given a sequence of twelve digits between 0 and 9, say $\overline{a_1a_2\dots a_{11}a_{12}}$, we need to consider the following sum modulo 10

$$\sum_{k=1}^{12}(2+(-1)^k)a_k = a_1 + 3a_2 + a_3 + 3a_4 + \cdots + a_{11} + 3a_{12}. \tag{7}$$

The sum needs to agree with the last digit for $\overline{a_1a_2\dots a_{11}a_{12}a_{13}}$ to be a valid 13-digit ISBN.

Problem 5. Check whether the following sequence is a valid 13-digit ISBN.

(a) 9780439139609

(b) 9780439784542

(c) 9781178050237

(d) 9783110014635

Problem 6. Try Problem 4 for 13-digit ISBN. Compare the results with those of Problem 4.

Problem 7. For the 13-digit ISBN, modify the error-detection mechanism, as best as you could, so that it can detect transposition of neighboring digits, in addition to detecting other simple errors.

2 Error Correction

From the exercises above, we see that both the 10-digit and 13-digit ISBN can detect single-digit error. However, it cannot correct the error automatically upon discovering it, since any digit could be the one with error. In fact, early computers do not have such capability and would stop if it runs into an error while reading codes.

Ingenuity is a robotic helicopter that has recently taken flight on Mars. Due to its distance from Earth, there is at least a 3 minute delay in information transfer to Mars. The code received by Ingenuity can contain errors due to random disturbances of electronic devices or celestial bodies. Instead of detecting an error and waiting for a response from Earth, Ingenuity implements advanced error correcting code to fix many problems immediately.

In this section, we will develop many early examples of error correcting codes. For simplicity, we only use the digits, or bits, 0 and 1 in a string. We also assume that all codewords in a code have the same length.

2.1 Repeating code

The simplest idea is to repeat each bit several times. For example, if we have the digits 101 and encode it by repeating each digit 3 times, then we have the codeword

111000111

Suppose we see 111000110 instead, then we can immediately fix the error and recover the original string of digits 101. Not only does this method detect the existence of single-digit and transposition errors, but also their locations and hence fix them.

Definition 1. Suppose we fix an integer k to be the length of the string and r the number of times a digit is to be repeated. Then the total length of the codeword is $n = kr$. All such possible codewords together is called a ***repeating*** $[n, k]$***-code***.

For example, a repeating $[4, 2]$-code contains the codewords $0000, 0011, 1100$ and 1111.

To measure the efficiency of codes, we can use the information rate defined by

$$R = \frac{\log_2(w)}{n}, \tag{8}$$

where w is the total number of codewords in the code, and n is the length of each codeword. Note that $w = 2^k$ so the numerator, $\log_2(w)$, represents the number of bits in the original code. The higher the R value, the more efficient the code is.

The information rate of the repeating $[4, 2]$-code is $\frac{\log_2(4)}{4} = \frac{1}{2}$.

Problem 8. Write down all the codewords in a repeating $[6, 2]$-code and calculate its information rate.

Problem 9. What is the information rate of a repeating $[n, k]$-code in general?

Problem 10. (a) Can you come up with a repeating $[4, 2]$-code for which there is no way to correct transposition error?

(b) Prove that, when $r > 2$, a repeating $[n, k]$-code can always detect and correct single-digit and transposition errors.

2.2 Hamming's square code

An example of a more complicated coding scheme is Hamming's square code. Begin with a message that has 4 bits. First, write this message in a 2×2 square. Compute the sum of each row and write it at the end of each row. Compute the sum of each column and write it at the bottom of the column. Finally compute the sum of all entries and write it in the lower right corner to complete a 3×3 square. Reading out the 9 bits then gives the codeword. Since we only have 0s and 1s in the alphabet, the addition rules will be

$$0 + 0 = 0, 1 + 0 = 1, 1 + 1 = 0.$$

For example, if the message is 1011, then the codeword is 101110011.

$$1011 \longrightarrow \begin{array}{cc} 1 & 0 \\ 1 & 1 \end{array} \longrightarrow \begin{array}{cc|c} 1 & 0 & 1 \\ 1 & 1 & 0 \\ \hline 0 & 1 & \end{array} \longrightarrow \begin{array}{cc|c} 1 & 0 & 1 \\ 1 & 1 & 0 \\ \hline 0 & 1 & 1 \end{array} \longrightarrow 101110011$$

Problem 11. The following codewords are encoded using the method above. Correct any single-digit or transposition error if there is any.

(a) 110110011

(b) 100101011

(c) 001010110

Problem 12. (a) Prove each row and column of the 3×3 table sums to 0 (mod 2).

(b) If a single-digit error is present in the codeword, can it be discovered and fixed?

(c) If a transposition error is present in the codeword, can it be discovered and fixed?

Problem 13. (a) Show that the information rate of Hamming's 2×2 square code is $\frac{4}{9}$.

(b) Generalize this coding method to $m \times n$ rectangular code. Find its information rate.

(c) If mn is fixed, what values of m and n give the information rate maximum? Compare it to the maximal information rate of the repeating code.

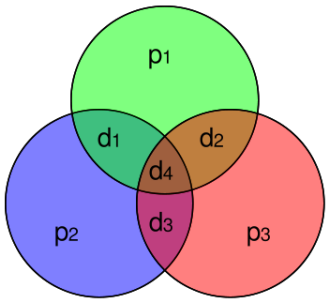
Problem 14. Is the $m \times n$ rectangular code capable of correcting single-digit errors or transposition errors?

Problem 15. If the last digit in the codeword is removed, can we still decode the message? What about fixing single-digit error or transposition error? What is the information rate?

From Problem 13, we know that the $m \times n$ rectangular code is much more efficient at encoding information than the simple repeating code. In fact, the information rate increases as the length of the message increases.

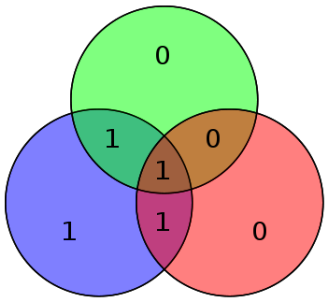
2.3 Hamming's [7,4]-code

For a message word with 4-bits, we will describe another encoding mechanism, which has higher information rate than Hamming's 2×2 square code. It is called Hamming's $[7, 4]$-code. Suppose we have the 4-bit word $\overline{d_1 d_2 d_3 d_4}$. Define three bits $p_1, p_2, p_3 \in \{0, 1\}$ such that four bits in the same circle add up to 0 in the diagram below.



The codeword is then $\overline{p_1 p_2 d_1 p_3 d_2 d_3 d_4}$.

For example, if the bits are 1011, then the diagram above becomes



So $p_1 = 0, p_2 = 1, p_3 = 0$ and the codeword is 0110011.

Suppose you are given a 7-bit message $\overline{a_1 a_2 \dots a_7}$. Define the check bits c_1, c_2, c_4 by

$$\begin{aligned} c_1 &= a_1 + a_3 + a_5 + a_7, \\ c_2 &= a_2 + a_3 + a_6 + a_7, \\ c_4 &= a_4 + a_5 + a_6 + a_7. \end{aligned}$$

Problem 16. Show that $\overline{a_1 a_2 \dots a_7}$ is a codeword if and only if $c_1 = c_2 = c_4 = 0$.

Amazingly, if $\overline{a_1 a_2 \dots a_7}$ is different from a codeword by a single digit, the binary number $c_4 c_2 c_1$ gives the location of error digit. That is why the order $\overline{p_1 p_2 d_1 p_3 d_2 d_3 d_4}$ is not arbitrary.

Problem 17. (a) List all the codewords in Hamming's $[7, 4]$-code.

(b) What is the information rate? How does it compare to the information rate of Hamming's 2×2 square code?

Problem 18. Pick your favorite 3-digit integer in decimal. Convert it to binary number and break it into blocks of 4 bits (add appropriate number 0 to the front to make the number of bits divisible by 4). Then encode each block using Hamming's $[7, 4]$-code.

Problem 19. Pick a codeword and change a single digit. Calculate the check bits c_1, c_2 and c_4. Treat $c_4 c_2 c_1$ as a 3-digit binary number and convert it to decimal. Do this a few times and what pattern do you notice? Can you prove it?

Problem 20. (a) Generalize Hamming's $[7, 4]$-code to encoding messages with 8 bits.

(b) Can you generalize Hamming's $[7, 4]$-code to encoding messages with 2^k bits for $k \geq 2$? What is the length of the final code?