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 number. 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 number 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

$$\sum_{k=1}^{9} k a_k = 1 \cdot 0 + 2 \cdot 3 + 3 \cdot 1 + 4 \cdot 6 + 5 \cdot 1 + 6 \cdot 6 + 7 \cdot 0 + 8 \cdot 1 + 9 \cdot 7 \tag{3}$$

$$= 145 \tag{4}$$

$$\equiv 2 \pmod{11}. \tag{5}$$

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

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

(a) 0439023521

(b) 311001436X

(c) 0439784542

Solution. (a) The sum is 1 modulo 11 with last digit 1. This is a valid ISBN number.

(b) The sum is 10 modulo 11 and the last digit is 10. This is a valid ISBN number.

(c) The sum is 9 modulo 11 and the last digit is 2. This is not a valid ISBN number.

□

Problem 2. Given a sequence $\overline{a_1 a_2 \dots a_{10}}$, 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 $\overline{a_1 a_2 \dots a_{10}}$ is the number in Problem 1(a), 1(b), and 1(c) respectively?

Solution. (a) The remainder is 0.

(b) The remainder is 0.

(c) The remainder is 4.

□

Problem 3. (a) If the sequence $\overline{a_1 a_2 \dots a_{10}}$ is a valid ISBN number, show that the sum in (6) is divisible by 11.

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

Solution. (a) Note that $k \equiv 11 - k \pmod{11}$. Since the sequence is a valid ISBN number, we can write

$$\begin{aligned} 10 \cdot a_1 + 9 \cdot a_2 + \cdots + 2 \cdot a_9 + a_{10} &\equiv -a_1 - 2a_2 - \cdots - 9a_9 + a_{10} \pmod{11} \\ &= -\left(\sum_{k=1}^{9} k a_k \right) + a_{10} \\ &\equiv 0 \pmod{11}. \end{aligned}$$

(b) We can proceed in reverse to obtain $10 \left(\sum_{k=1}^{9} k a_k \right) \equiv 10 a_{10} \pmod{11}$. Since 10 and 11 are relatively prime, we can cancel the factor of 10 and the sequence is a valid ISBN number.

□

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

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

(c) What happens to a valid ISBN number if you swap two adjacent digits?

Solution. (a) If you change the kth digit for $1 \leq k \leq 9$, then the sum $\sum_{k=1}^{9} ka_k$ will change by mk for $0 \leq m \leq 10$. Since k and 11 are relatively prime, the only multiples of k that are 0 modulo 11 are those multiples of 11. Thus the ISBN number will not be valid. If you change the 10th digit, then the ISBN number will not be valid.

(b) Take for example Problem 1(c), which is not a valid ISBN number. Change the first digit to a 4 to obtain a valid ISBN number.

(c) Let $\overline{a_1a_2\ldots a_{10}}$ a valid ISBN number. Swap adjacent digits a_i and a_{i+1}. We change the sum by adding a copy of a_i and subtracting a copy of a_{i+1}. For the ISBN number to remain valid, we need $a_i - a_{i+1}$ to be congruent to 0 modulo 11. However, $0 \leq a_i, a_{i+1} \leq 10$ so $-10 \leq a_i - a_{i+1} \leq 10$. The ISBN number only remains valid when $a_i = a_{i+1}$ and we obtain the same ISBN number by swapping.

□

Besides the 10-digit ISBN number, the 13-digit ISBN number is also in use. For example, the 13-digit ISBN of the fourth Harry Potter book is 9780439139601, compare to the 10-digit ISBN number 0439139600. The first three digits of the 13-digit ISBN number 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 $\overline{a_1a_2\ldots a_{11}a_{12}}$, 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 $\overline{a_1a_2\ldots a_{11}a_{12}a_{13}}$ to be a valid 13-digit ISBN number.

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

(a) 9780439139609

(b) 9780439784542

(c) 9781178050237

(d) 9783110014635

Solution. (a) The sum in (7) is congruent to 9 modulo 10. This is a valid 13-digit ISBN number.

(b) The sum in (7) is congruent to 0 modulo 10. This is not a valid 13-digit ISBN number.

(c) The sum in (7) is congruent to 7 modulo 10. This is a valid 13-digit ISBN number.

(d) The sum in (7) is congruent to 9 modulo 10. This is not a valid 13-digit ISBN number.

□

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

Solution. (a) Let $\overline{a_1a_2\dots a_{11}a_{12}a_{13}}$ be a valid 13-digit ISBN number. Take $1 \le k \le 12$ and change a_k to b_k. Since $0 \le a_k, b_k \le 9$, we have $-9 \le a_k - b_k \le 9$. When k is odd, the new ISBN number will not be valid. When k is even, 3 and 10 relatively prime implies that $3(a_k - b_k)$ will not be a multiple of 10 for $-9 \le a_k - b_k \le 9$ when $a_k \ne b_k$. If a_{13} is changed, then the ISBN number is no longer valid.

(b) Take for example Problem 5(b), which is not a valid ISBN number. Change the last digit to 0 to obtain a valid ISBN number.

(c) Take for example Problem 5(c). If we swap 0 and 5 to obtain 9781178500237, the new ISBN number is still valid. In general, if we swap two digits a_i and a_{i+1} where $1 \le i \le 11$, then we change the sum in (7) by $2a_{i+1} - 2a_i$. The sum is the same modulo 10 when $a_{i+1} - a_i \equiv 0 \pmod 5$. This result differs from that of the 10-digit ISBN number.

□

Problem 7. For the 13-digit ISBN number, 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.

Solution. We could assign ISBN numbers so that no two adjacent digits in the first twelve digits have a difference of 5. □

2 Error Correction

From the exercises above, we see that both the 10-digit and 13-digit ISBN numbers 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.

Solution. The possible codes are 000000, 000111, 111000, and 111111. The information rate is $R = \frac{\log_2(4)}{6} = \frac{1}{3}$. □

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

Solution. This is just combinatorics. The number of possible codewords is 2^k. Thus the information rate is

$$\frac{\log_2\left(2^k\right)}{n} = \frac{k}{n} = \frac{1}{r}.$$

□

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?

Solution. (a) The code 0101 could be 0000, 1111, 0011, or 1100.

(b) When $r > 2$, a single-digit error or transposition can affect at most one element of each repeated string. By taking the most common digit of each repeated portion, we can detect and correct any error.

□

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.

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

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

Solution. (a) 101110011 or 110101011

(b) 110101011

(c) 000110110

□

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?

Solution. (a) We will focus on rows, but the same argument works for column sums. Take one of the first two rows with entries a_1, a_2, and a_3. Then $a_1 + a_2 = a_3$ so $a_1 + a_2 + a_3 = 2a_3 \equiv 0 \pmod 2$. In the last row, we have b_1, b_2, and b_3 where b_3 is the sum of the three first row elements, the three second row elements, b_1, and b_2. The first two rows sum to zero so $b_1 + b_2 = b_3$. Then $b_1 + b_2 + b_3 \equiv 0 \pmod 2$.

(b) If a single-digit error is present, there will be a row and column sum that is no longer zero. We can fix the error since the corresponding row and column in the grid is where the single-digit error occurs.

(c) If a transposition error is present within a row of the grid, two adjacent columns will no longer sum to 0 (mod 2). If a transposition error is present between two rows of the grid, two rows and two columns will no longer sum to 0 (mod 2). As in Problem 11(a), there could be multiple ways to correct the transposition.

□

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.

Solution. (a) This is just combinatorics. We can choose any 4 digits for the original message. All other digits of the codeword are determined. Thus $w = 2^4$ so the information rate is $R = \frac{4}{9}$.

(b) We will take row sums, column sums, and entry sums as in the 2×2 case to produce a string of length $(m + 1)(n + 1)$. By a similar argument to (a), the information rate is $R = \frac{mn}{(m+1)(n+1)}$.

(c) The information rate $R = \frac{mn}{mn+m+n+1}$ by part (b). To maximize this value, we want to minimize $m+n$ while keeping mn fixed. Finding all pairs of factors of mn, take the pair with the smallest difference.

For a repeating code of length mn, the information rate is $R = \frac{1}{r}$ for r the number of repetitions of each digit by Problem 9. Thus the maximal information rate of Hamming's square code is higher than the information rate of the repeating code.

□

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

Solution. The $m \times n$ rectangular code can correct single-digit error as in Problem 12(a). However, Problem 11(a) shows that the code cannot correct 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?

Solution. Without the final digit, we can still decode the message by filling in the the table except for the bottom right entry.

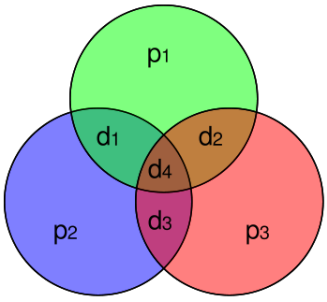
We will be able to detect an error, but there are cases where we won't be able to determine which type of error it is. For instance, take the code 101110011 from the beginning of the section. Remove the last digit. We can transpose the second and third digits to obtain 11011001. Only the second column sum is not correct. We can fix this by transposing the second and third digits back or by switching the final digit.

The information rate is $R = \frac{mn}{(m+1)(n+1)-1}$, which is slightly larger than the information rate of the code with the extra digit. □

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.

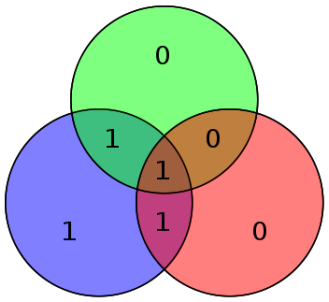
2.3 Hamming's [7,4]-code

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 $c_1 = c_2 = c_4 = 0$.

Solution. ($\Rightarrow$) Assume that $\overline{a_1 a_2 \dots a_7}$ is a codeword. Then

$$\begin{aligned} c_1 &= a_1 + a_3 + a_5 + a_7 \\ &= p_1 + d_1 + d_2 + d_4 \\ &= (d_1 + d_2 + d_4) + d_1 + d_2 + d_4 \\ &\equiv 0 \pmod 2 \\ c_2 &= a_2 + a_3 + a_6 + a_7 \\ &= p_2 + d_1 + d_3 + d_4 \\ &= (d_1 + d_3 + d_4) + d_1 + d_3 + d_4 \\ &\equiv 0 \pmod 2 \\ c_4 &= a_4 + a_5 + a_6 + a_7 \\ &= p_4 + d_2 + d_3 + d_4 \\ &= (d_2 + d_3 + d_4) + d_2 + d_3 + d_4 \\ &\equiv 0 \pmod 2. \end{aligned}$$

($\Leftarrow$) Assume that $c_1 = c_2 = c_4 = 0$. Then a_1 satisfies the necessary equation for p_1, a_2 satisfies the necessary equation for p_2, and a_4 satisfies the necessary equation for p_4. □

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?

Solution. (a) List out the 16 different 4-bit words in binary counting order. We obtain the following 16 correct codewords 0000000, 11001001, 0101010, 1000011, 1001100, 0100101, 1100110, 0001111, 1110000, 0011001, 1011010, 0110011, 0111100, 1010101, 0010110, and 1111111.

(b) The information rate is $R = \frac{4}{7}$. This is higher than Hamming's 2x2 square code and any of the repeating codes.

□

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.

Solution. We try the number 144. Then its binary expansion is 10010000. We have two blocks of 1001 and 0000 to obtain 0011001 followed by 0000000. □

Problem 19. Pick a codeword and change a single digit. Calculate the check bits c_1, c_2 and c_4. Treat $c_4c_2c_1$ 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?

Solution. We try the codeword 0010110 which encodes the message 1110. Switch the second digit to obtain 0110110. Then the check bits are $c_1 = 0$, $c_2 = 1$, and $c_4 = 0$. The 3-digit binary number 010 represents 2 in decimal. This is the location of the error.

Note that c_1 is the sum of all a_i for which i has a 1 in its binary expansion. Similarly, c_2 is the sum of all a_i for which i has a 2 in its decimal expansion. Finally, c_4 is the sum of all a_i for which i has a 4 in its binary expansion. Switching the kth digit of the code will change the parity of c_1 only if there is a 1 in the binary expansion of k, c_2 only if there is a 2 in the binary expansion of k, and c_4 only if there is a 4 in the binary expansion of k. Thus $c_4c_2c_1$ will be the binary representation of k, the location of the error. □

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?

Solution. (a) Let $d_1d_2d_3d_4d_5d_6d_7d_8$ be the original 8-bit word. We will need 4 check digits to represent the 8+4 possible positions in our final code. Denote the final code by

$$a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}a_{12}.$$

By writing 1 through 12 in binary, we want to define the check digits by

$$c_1 = a_1 + a_3 + a_5 + a_7 + a_9 + a_{11}$$

$$
\begin{aligned}
c_2 &= a_2 + a_3 + a_6 + a_7 + a_{10} + a_{11} \\
c_3 &= a_4 + a_5 + a_6 + a_7 + a_{12} \\
c_4 &= a_8 + a_9 + a_{10} + a_{11} + a_{12}
\end{aligned}
$$

so that $c_4c_3c_2c_1$ represents the position of an error in the code. Note that a_1, a_2, a_4, and a_8 only appear once. These will be represent the added bits p_1, p_2, p_3, and p_4 respectively. The code will be

$$p_1p_2d_1p_3d_2d_3d_4p_4d_5d_6d_7d_8$$

where the p_i are defined as

$$
\begin{aligned}
p_1 &= a_3 + a_5 + a_7 + a_9 + a_{11} \\
&= d_1 + d_2 + d_4 + d_5 + d_7 \\
p_2 &= a_3 + a_6 + a_7 + a_{10} + a_{11} \\
&= d_1 + d_3 + d_4 + d_6 + d_7 \\
p_3 &= a_5 + a_6 + a_7 + a_{12} \\
&= d_2 + d_3 + d_4 + d_8 \\
p_4 &= a_9 + a_{10} + a_{11} + a_{12} \\
&= d_5 + d_6 + d_7 + d_8
\end{aligned}
$$

in order to guarantee all of the check bits are 0 in a correct code.

(b) The process is much the same as part (a), but it requires somewhat complicated indexing. We will end up with codes of length $2^k + k + 1$.

□