NUMB3RS Activity: Error Correction
Episode: "In Plain Sight"

Topic: Error Correction Codes **Grade Level**: 9 - 12
Objective: Students will learn how Hamming codes are used for error correction.
Time: 30 minutes

Introduction

Most of the information in the world today is digital. Televisions, CDs, and computers store and display information digitally, but these devices sometimes misread pieces of the digital data, resulting in poor pictures or lost information. To make sure that computers read data correctly, many error correction techniques have been developed. These techniques help a computer to identify and fix mistakes. In the episode "In Plain Sight", Charlie and Larry give a brief introduction to Reed-Solomon error correcting code. Reed-Solomon is extremely complex, so a simpler error correcting code called Hamming code is used here.

Richard Hamming's code is dependent upon the concept of *parity*. Parity indicates if the number of 1s in a piece of data is even or odd. The parity is even if there is an even number of 1s, and odd if there is an odd number of 1s (i.e. the binary sequence 101101000110 has even parity because there are six 1s in the sequence.)

Discuss with Students

***NUMB3RS* Example** During the episode "In Plain Sight", Larry scratches a CD with a paper clip. He tells Don "If I scratched a record, it would be ruined, but this CD will still play. I've created gaps in the readable data." If data has been destroyed, how can the CD still be played? Charlie explains: "[With] Reed-Solomon Error Correcting Code, the software fills in the gaps, making an educated guess about what it should be—resulting in complete images from partial data."

The Hamming error correcting code takes a piece of code and makes a new one by adding *parity bits*, designated by P_1, P_2, P_4, P_8, etc. Note that parity bits are numbered according to the powers of 2. These numbers show where to put each parity bit.

Each parity bit "checks" certain numbers in a sequence: For example, for the code:

1011010

- P_1 checks every odd-numbered position. **1**0**1**1**0**1**0**
- P_2 checks positions 2, 3, 6, 7, 10, 11, …, alternately reading two and then skipping two. 1**01**10**10**
- P_4 checks positions 4, 5, 6, 7, 12, 13, 14, 15, …alternately reading four in a row and then skipping four in a row. 101**1010**

Example Say you want to encode the sequence “1 0 1 1” as a Hamming sequence with even parity.

The first step is to insert the parity bits into the correct place. You will make a new sequence. Remember that the subscript of the parity bit tells you where to put it:

$$\mathbf{P_1\ \ P_2\ \ 1\ \ P_4\ \ 0\ \ 1\ \ 1}$$

P_1 goes first, followed by P_2 and so on. There’s no third parity bit, so you would put in the first number from the original sequence, followed by P_4, which should of course be the fourth number in the new sequence. There are no fifth, sixth, or seventh parity bits, so the last three numbers come from the original sequence. The next step is to replace the parity bits with numbers, 0 or 1.

On the previous page, we said that P_1 checks the odd-numbered positions: $\underline{\mathbf{P_1}}\ \ P_2\ \ \underline{\mathbf{1}}\ \ P_4\ \ \underline{\mathbf{0}}\ \ 1\ \ \underline{\mathbf{1}}$. Since we are using even parity, the odd-numbered positions must add up to an even number. That means that P_1 must equal 0, since $P_1 + 1 + 0 + 1 = 2$. So our sequence so far is: $0\ \ P_2\ \ 1\ \ P_4\ \ 0\ \ 1\ \ 1$.

Next, we substitute a number for P_2. We know which positions P_2 checks, so we identify them: $1\ \ \underline{\mathbf{P_2\ 1}}\ \ P_4\ \ 0\ \ \underline{\mathbf{1\ \ 1}}$. We’re still using even parity, so these numbers must add up to an even number. Again, this requires that $P_2 = 1$. So our sequence so far is: $0\ \ 1\ \ 1\ \ P_4\ \ 0\ \ 1\ \ 1$.

We do the same thing for P4. Identify which places it checks: $1\ \ 1\ \ 1\ \ \underline{\mathbf{P_4\ 0\ \ 1\ \ 1}}$, then figure out what P_4 must equal. In this case, again, $P_4 = 0$. So, writing the original sequence 1 0 1 1 as a Hamming sequence gives us:

$$\mathbf{0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1}$$

Now, suppose that this data is misread as 0 1 1 0 0 1 0. How would the error be detected? The Hamming code will catch the error by recalculating the parity bits:

$$\underline{\mathbf{0}}\ \ 1\ \ \underline{\mathbf{1}}\ \ 0\ \ \underline{\mathbf{0}}\ \ 1\ \ \underline{\mathbf{0}}$$

The underlined digits checked by P_1 add up to an odd number. There must be an error in position 1, 3, 5, or 7.

$$0\ \ \underline{\mathbf{1}}\ \ \underline{\mathbf{1}}\ \ 0\ \ 0\ \ \underline{\mathbf{1}}\ \ \underline{\mathbf{0}}$$

The underlined digits checked by P_2 add up to an odd number. There must be an error in position 2, 3, 6, or 7.

$$0\ \ 1\ \ 1\ \ \underline{\mathbf{0}}\ \ \underline{\mathbf{0}}\ \ \underline{\mathbf{1}}\ \ \underline{\mathbf{0}}$$

The underlined digits checked by P_4 add up to an odd number. There must be an error in position 4, 5, 6, or 7. If there is at most one error, then it must be in position 7.

Student page answers: **1.** *$P_1 = 1$ because the parity of {0 0 1} is odd, $P_2 = 0$ because the parity of {0 1 1} is even, $P_4 = 0$ because the parity of {0 1 1} is even* **2.** *error in 3rd position* **3.** *error in 4th position* **4.** *No error* **5.** *$P_1 = 0$ $P_2 = 1$ $P_4 = 1$ $P_8 = 1$ $P_{16} = 1$, $P_{32} = 0$.*

Name: ______________________________ Date: _______________

NUMB3RS Activity: Error Correction

In the episode "In Plain Sight", Larry scratches a music CD but says it will still play perfectly because of the error correction code on the CD. How does this work?

Correction codes depend upon an idea called *parity.* Parity indicates if the number of 1s in a piece of data is even or odd. The parity is even if there is an even number of 1s, and odd for an odd number (i.e. the sequence 11110 has even parity because there is an even number (4) of 1's in the sequence.) For this activity, we will use even parity.

A music CD is an example of digital information. Suppose a very small piece of music data is represented by the sequence of bits 0 1 1 0. To use "Hamming error correcting code," parity bits are inserted in the data sequence to check if the data is being read accurately. These parity bits (P_1 P_2 P_4 P_8…) are inserted at locations 1, 2, 4, 8, etc. (powers of 2) in the transmitted data to check the parity of a subsequence of the original data, starting with itself:

- P_1 checks one bit, skips one bit, checks one bit, etc. (positions 1, 3, 5, etc.)
- P_2 checks 2 bits, skips 2 bits, checks 2 bits, skips 2 bits, etc. (positions 2, 3, 6, 7, etc.)
- P_4 checks 4 bits, skips 4 bits, checks 4 bits, skips 4 bits, etc. (positions 4, 5, 6, 7, 12, 13, 14, 15, etc.)

The parity bit is a 1 or a 0 to make the number of 1s in the bits it checks even.

Example Code the sequence 0 1 1 0 into Hamming error correction code.

Step 1: Insert parity bits (P_1 P_2 P_4) at the 1st, 2nd , and 4th data positions in the list.
$P_1\ \ P_2\ \ 0\ \ P_4\ \ 1\ \ 1\ \ 0$

Step 2: Determine the value of the 1st parity bit P_1.
P_1 checks the sequence $\underline{P_1}\ \ P_2\ \ \underline{0}\ \ P_4\ \ \underline{1}\ \ 1\ \ \underline{0}$

$P_1 + 0 + 1 + 0$

To make even parity, $P_1 = 1$.

Step 3: Determine the value of the 2nd parity bit P_2.
$1\ \ P_2\ \ 0\ \ P_4\ \ 1\ \ 1\ \ 0$
P_2 checks the sequence $1\ \ \underline{P_2\ \ 0}\ \ P_4\ \ 1\ \ \underline{1\ \ 0}$
$P_2 + 0 + 1 + 0$

To make even parity, $P_2 = 1$.

Patrick Flynn, Turner High School, Kansas City, KS

Step 3: Determine the value of the 3rd parity bit P_4.

1 1 0 P_4 1 1 0

P_4 checks the sequence 1 1 0 $\underline{P_4 \;\; 1 \;\; 1 \;\; 0}$

$P_4 + 1 + 1 + 0$

To make even parity, $P_4 = 0$.

The data sequence is now coded with Hamming error correction code as: 1 1 0 0 1 1 0

What makes Hamming extremely useful is its ability to correct misread data. Suppose the data from the example is misread as 0 1 1 0 0 0 0. How is the error caught and fixed?

The Hamming code can correct a single error like this by using the parity bits. It reads and records the positions in which the digits are wrong.

For the misread sequence 0 1 1 0 0 0 0, P_1 fails because $0 + 1 + 0 + 0 = 1$, which is odd. But, it shouldn't be odd. P_2 passes, and P_4 passes. The 1st digit must be the source of the error and its value is switched.

Exercises

1. Determine the parity bit values for the Hamming sequence P_1 P_2 0 P_4 0 1 1
P_1 = ______ P_2 = ______ P_4 = ______

Use the Hamming error correction code to spot the error (if there is one.)

2. 0 1 0 1 1 0 0 ____

3. 0 1 1 1 0 1 1 ____

4. 1 0 1 0 1 0 1 ____

5. Music CDs use strings of 32 bits, including error correction. Determine the parity bit values for the CD data sequence:

P_1 P_2 1 P_4 1 0 1 P_8 0 0 1 1 1 0 0 P_{16} 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0 P_{32}

P_1 = _____ P_2 = _____ P_4 = _____ P_8 = _____ P_{16} = _____ P_{32} = _____

The goal of this activity is to give your students a short and simple snapshot into a very extensive math topic. TI and NCTM encourage you and your students to learn more about this topic using the extensions provided below and through your own independent research.

Extensions

Activity: Coded numbers in your house.

Introduction

While Hamming codes auto-correct mistakes due to misreads, they require many extra digits to be added to the data. For small data lists (like UPC codes on grocery items, credit card numbers, ISBN numbers on textbooks, etc.) it is much easier to just rescan the data list if it is read wrong. The computer determines if the data was read wrong by the use of an extra digit (called a check digit). Below is an example.

UPC codes

0 12345 02995 0

All items bought in a store have a Universal Product Code (UPC) on the label. The UPC code to the left has the sequence 0 1 2 3 4 5 0 2 9 9 5 0. When the computer scans this item at the cash register, it performs a calculation to check if it was scanned correctly. The answer to this calculation must be the same as the last digit in the sequence (called the check digit). If the computer doesn't get the correct answer, it is told to scan again.

Here is how it works…Take a UPC code (without the check digit) and multiply each single digit in the following way…

$$\begin{array}{cccccccccccc} & 1 & 2 & 3 & 4 & 5 & 0 & 2 & 9 & 9 & 5 \\ \text{X} & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 \\ \hline & 3 & 2 & 9 & 4 & 15 & 0 & 6 & 9 & 27 & 5 \end{array}$$

Now add the results 3 + 2 + 9 + 4 +15 + 0 + 6 + 9 + 27 + 5 = 80
The check digit is added to the sum and the result must be divisible by 10.
In this case, 80 + 0 is divisible by 10, so the check works.

Additional Resources

Visit Joe Gallian's web site that calculates the check digit for many other number sequences: **http://www.d.umn.edu/~jgallian/fapp5/**

For the Student

- So far you've seen how to detect single errors. Research how two-digit errors are detected using Hamming error correction code.
- Use your calculator to perform Hamming error correction by multiplying matrices. Research Hamming (7, 4) to find out how.
- Research other error correction codes, such as repetition, two-out-of-five, and Reed-Solomon error correction code.

Related Topic:

Find out how CD players read digital data from a disk:
http://www.howstuffworks.com/cd.htm

Patrick Flynn, Turner High School, Kansas City, KS