MITOCW | MIT6_004S17_01-02-12-05_300k

We will now review a problem on error correction and detection in order to better understand how the selected encoding of a message can help both detect and correct transmission errors.

In this problem messages consist of 9 data bits and 7 parity bits.

Each Dij represents a data bit which belongs to row i and column j.

The Pij bits are used to make the parity of each row and column be odd.

The Pxx box, which is in the bottom right corner, is used to make the parity of the entire message odd.

In other words, it makes the total number of ones be odd.

The first question we want to ask ourselves is what is the minimum Hamming Distance between valid code words using this encoding?

In order to figure that out, we need to think about how many bits of the encoding are changed if I want to change one of the data bits.

The easiest way to see this is through an example.

Suppose that I want to change D01 from a 0 to a 1.

The first thing changing is my data bit itself.

Then P0x would get flipped because I just added another 1 to my row which means that I need to flip the parity bit in order for the total number of 1's in my row to remain odd.

The next change is that Px1 needs to get flipped in order to maintain an odd parity in my column.

So far, 3 entries have changed which implies that the parity of the entire message flipped.

In order to keep the parity of the entire message odd, Pxx needs to be flipped as well so that there are a total of 4 entries being flipped which will cause the odd parity of the entire message to remain unchanged.

This means that our minimum hamming distance for this encoding equals 4 because any time we flip a data bit, a total of 4 entries need to change to maintain our encoding.

We know that to detect an E-bit error, the hamming distance (HD) has to be greater than E, or in other words, the HD >= E + 1.

So to detect a 1-bit error, one needs a HD >= 2.

We also know that to correct an E-bit error, the HD > 2E, or the HD >= 2E + 1.

So to correct a 1-bit error, one needs to have a HD >= 3.

For this encoding, since our hamming distance equals 4, that means we should be able to both detect and correct 1-bit errors.

Taking a look at this message, if we check the parity of each row, column, and full message, we see that in all cases our parity is odd.

Since odd parity indicates a valid message, there are no errors in this message.

Looking at this message we see that all the row parities are odd, but the parity of column 1 is even.

This means that there is an error in column 1.

However, since there were no errors in the parity of the rows, this means that the bit in error is the parity bit itself.

If we flip Px1 from a 0 to a 1 we now have a valid message again.

Looking at the parity of each row and column in this message, we see that row 0 has an even parity, and column 0 has an even parity.

This means that the data bit at row 0, column 0 is wrong.

If we flip this bit, we now see that all of our row and column parities are correct, and our total message parity is correct as well, so we have restored our message by identifying the bit in error.

In this example, all of the row parities and column parities are odd.

However, the parity of the entire message is even.

This means that the bit in error in this message is the Pxx bit itself.

Flipping that bottom right bit results in a valid message once again.

Now let's go back to our original valid message and see if we can correct 2-bit errors.

Suppose we flip D11 to 0 and D22 to 1 to create a 2-bit error.

Now when we look at the parity of each row, we find that both row 1 and 2 have a parity error.

Similarly, columns 1 and 2 have a parity error.

The total message parity remains correct.

Given this information, we know that an error occurred in the transmission of this message, but we cannot identify exactly which bits are in error because there is more than one possible way to alter two bits and arrive at a valid message.

This demonstrates that we can detect a 2-bit error, but we can't correct a 2-bit error.

Going back to our original claims about hamming distances, this is in line with our expectations.

If E=2, meaning that we want to detect a 2-bit error, then the HD >= E+1 which equals 3.

To correct a 2-bit error, the HD >= 2E+1 which equals 5.

Since our hamming distance in this problem is 4, we can only detect 2-bit errors, but not correct them.