How does the Hamming code work?

Asked 16 years ago Modified 9 years, 7 months ago Viewed 19k times

15

When transmitting data, the Hamming code apparently allows you to recreate data that has been corrupted over the wire (an error correcting code).

How does this work and what are its limitations, if any?

Are there any better solutions for error correction (as opposed to retransmission)? Are there circumstances where retransmission is better?

communication hamming-code

Share

Improve this question

Follow

edited May 6, 2015 at 3:27

Joel

4,762 ● 9 ● 40 ● 55

asked Dec 23, 2008 at 10:42

paxdiablo

880k ● 241 ● 1.6k ● 2k

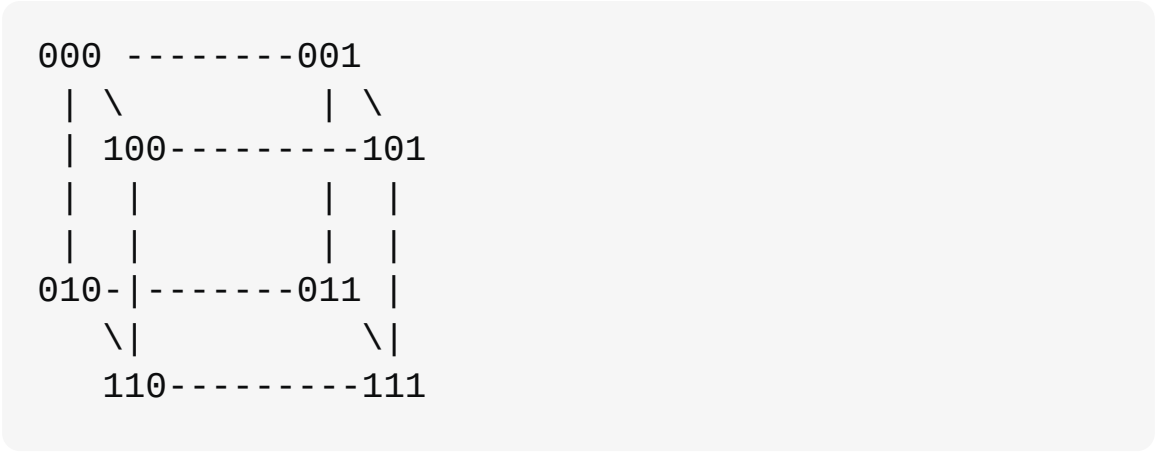
6 Answers

Sorted by: Highest score (default)

Let's try to explain it a bit:

40

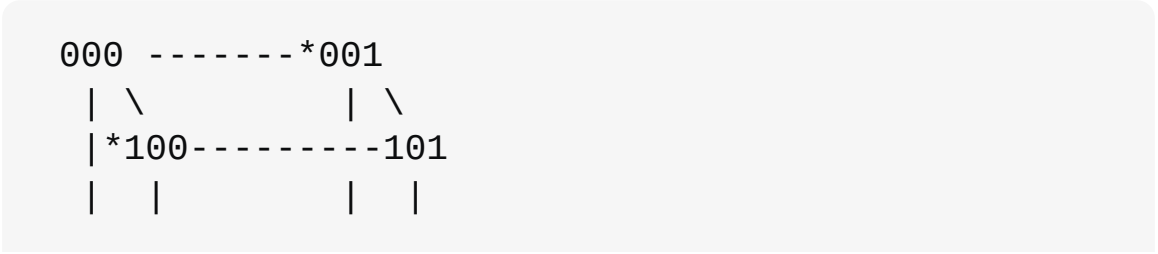
We have a 3 bit number. The possibilities can be presented as a cube where each bit represent an axis. The eight possibilities are on the corners.



Each defect (for example 101 is read as 100) results in a shift on a line on the cube.

If we use two bits for the data and the third for a parity check (say for example even parity). We lose 4 datapoints. But it has the advantage that we can detect a single bit failure (which transforms an even count of 1's into an odd count of ones). The odd numbers are marked with a *. And we see that each odd (wrongly transmitted) word is cornered by even (rightfully transmitted) words. So if we receive 100, we can be sure it is wrong.

But (with a single bit failure) the correct representation could have been 000, 101 or 110. So we can detect something has been wrong but we cannot detect what was wrong:



```
  |  |         |  |
*010-|-------011 |
    \|          \|
    110--------*111
```

This is called a one bit error detecting code.

If we use another bit for checking and thus remove one for the data. We are left with 1 databit and 2 "check bits". In this case, lets assume 000 and 111 are valid data representations and the other six are not. Now we have an interesting situation if one bit is mangled during transport. If we send 000 and receive 010, we see that 010 has one valid neighbor (000) and two invalid ones (110 and 011). So now we know that we intended to send 000 and are able to correct that:

```
 000 -------*001
  | \         | \
  |*100--------*101
  |  |         |  |
  |  |         |  |
*010-|------*011 |
    \|          \|
   *110---------111
```

This is called a one bit error correcting code.

Please note that a one bit error correcting code is also a two bit error detecting code.

And to put it more generally.

If you have n check bits, you have a n bit error detecting code. If you have 2n check bits, you have a n bit error correcting code.

Of course you should order the "valid" codes so that they do not border on each other.

Share Improve this answer Follow

edited Dec 23, 2008 at 11:48

answered Dec 23, 2008 at 11:28

Toon Krijthe

53.4k ● 38 ● 149 ● 202

That's a good explanation so +1 and very close to accepted. I can't imagine that you need 2 extra bits per bit to achieve this tho'. That would triple the data sent and make retransmissions a better solution. Or is your 1-bit case an edge case, and larger sizes use some lesser number of extra bits? – paxdiablo Dec 23, 2008 at 12:00

5 Hell, +1 for the diagrams alone. – Charlie Martin Dec 23, 2008 at 12:30

@Pax, as far as I know, it is the number of extra bits. So 5 bits with 2 check bits also make a 1 bit error correcting code. If I don't forget it I will check with the books at home tonight. – Toon Krijthe Dec 23, 2008 at 14:05

1 Briliant explanation, extra points for the diagrams. This answer is the missing link between mortals and the wikipedia page on Hamming codes. :-) Thanks! – Rolf Jan 4, 2009 at 22:38

6 The ASCII Art skills are strong with this one! – dj_segfault Mar 8, 2010 at 16:20

14

Here's the really high-level overview.

```
Suppose that every time I send a message, I send
thrie copies of the text.
Suppose that every time I send z message, I send
three copies of the teyt.
Suppose that every tyme I send a message, I send
three copies if the tezt.
```

By comparing characters and taking a simple majority vote in each position, you can correct single erroneous characters. However the cost of this scheme (amount of data that must be sent) is high, and it doesn't catch the unlikely-but-possible case of *two* errors in corresponding positions of different copies (as in the last word of the sample above).

Hamming codes (and other kinds of error-correcting codes, such as Reed-Solomon) are based on formulas that compute the extra data (rather than simple duplication). The added bits depend on combinations of the data bits in a way that errors in copying make detectable patterns of changes when the computation is repeated at the receiving end.

For sake of illustration, let's start with simple odd parity, adding a single bit to ensure that the total number of bits in a message is odd. So the message `10110110`

becomes `101101100` (five 1s, no extra 1 needed) and the message `10010110` becomes `100101101` (four 1s, extra 1 needed). If you receive a message of `101101101` and see that there are six 1s, you know that there's an error, but don't know where. Suppose we add more parity bits, each on depending only on a **part** of the message, as illustrated below by copying the bits considered and using '-' for bits ignored:

```
10110110
1-1-0-1- => 0
-0-1-1-0 =>  1
10--01-- =>   1
--11--10 =>    0
1011---- =>     0
----0110 =>      1
```

so the complete message is `10110110011001`. Now suppose a transmission error changes the third bit in the message, so that it reads `10010110011001`. When the receiver re-runs the error-checking computation, it fails to match:

```
10010110
1-0-0-1- => 1*
-0-1-1-0 =>  1
10--01-- =>   1
--01--10 =>    1*
1001---- =>     1*
----0110 =>      1
```

and the check bits that fail to match are exactly the ones affected by the third data bit. This is **not** a real, robust error-correction scheme; it's just a sketch to illustrate how

building in redundancy can help in identifying the exact nature of an error.

Share Improve this answer

Follow

answered Dec 23, 2008 at 13:58

joel.neely

30.9k ● 9 ● 57 ● 64

your answer is more understandable to me than the cube one. what would seal the deal is explaining how the errored bits are corrected. – Kent Fredric Dec 23, 2008 at 22:56

5

You will find details on the way it works here

More general information about Error Corecting Codes is available here

Share Improve this answer

Follow

answered Dec 23, 2008 at 10:45

Brann

32.3k ● 33 ● 120 ● 164

Thanks, Brann, I read the wikipedia article and it was pretty deep, technical-wise. I was hoping for a slightly more layman response that would serve as a basis for returning to wikipedia for the details. – paxdiablo Dec 23, 2008 at 10:58

3

Information about Hamming code is available here and here.

As for suitability , this explains why :

1. Error-correcting codes like Hamming are suitable for simplex channels where no retransmission can be requested.
2. Error-detection plus retransmission is often preferred because it is more efficient.

For comparison, consider a channel with error rate of per bit. Let block size be 1000 bits.

To correct a single error (by Hamming code), 10 check bits per block are needed. To transmit 1000 blocks, 10,000 check bits (overhead) are required. To detect a single error, a single parity bit per block will suffice. To transmit 1000 blocks, only one exter block (due to the error rate of per bit) will have to be retransmitted, giving the overhead of only 2001 (= 1000 + 1001) bits.

Share Improve this answer

Follow

answered Dec 23, 2008 at 11:05

Learning

8,175 ● 3 ● 38 ● 47

0

Hamming code is a mathematical trick to correct up to 4 lost bits in a serial transmission. It is also used in favour of the "parity bit" in modern memory chips.

Limitations are in the number of bits which can be restored, which is not more than 4. If more than 4 bits are lost, retransmission is required.

Different situations require different error correction techniques. Some of these techniques are listed in the

other posts here.

Share Improve this answer

Follow

answered Dec 23, 2008 at 11:22

Rolf

7,268 ● 6 ● 42 ● 57

4 lost bits per what? byte? n-bit value? – paxdiablo Dec 23, 2008 at 12:05

0

@GameCat and what about 2-bit error detecting code.

In this case lets say 111 has changed to 100. Then we can be sure that there are 2 bits wrong and we know this because the distance between 111 and 100 is 2 bits, and the distance of 000 and 100 is 1 bit. So if we know that there is 2-bit error we can be sure what is the right value.

Share Improve this answer

Follow

answered Mar 8, 2010 at 15:50

user288909

1