LE1.7

Bookmark this page

LE1.7.1: Parity

pontos 4 / 4 (sem classificação)

To protect stored or transmitted information one can add check bits to the data to facilitate error detection and correction. One scheme for detecting single-bit errors is to add a parity bit:

```
b0 b1 b2 ... bN-1 p
```

When using even parity, p is chosen so that the number of "1" bits in the protected field (including the p bit itself) is even; when using odd parity, p is chosen so that the number of "1" bits is odd. In the remainder of this problem assume that even parity is used.

To check parity-protected information to see if an error has occurred, simply compute the parity of information (including the parity bit) and see if the result is correct. For example, if even parity was used to compute the parity bit, you would check if the number of "1" bits was even.

If an error changes one of the bits in the parity-protected information (including the parity bit itself), the parity will be wrong, i.e., the number of "1" bits will be odd instead of even. Which of the following parity-protected bit strings has a detectable error assuming even parity?

11101101111011011

- (•) Has detectable error
- ( ) Appears okay

✔

11011110101011110

- ( ) Has detectable error
- (•) Appears okay

✔

10111110111011110

- (•) Has detectable error
- ( ) Appears okay

✔

00000000000000000

- ( ) Has detectable error
- (•) Appears okay

✔

Explanation

String #1 and #3 have an odd number of 1-bits, so since they don't have even parity, there has been a dete

Calculator

error.

Enviar

Answers are displayed within the problem

LE1.7.2: Error Detection

pontos 5 / 5 (sem classificação)

After finishing 6.004.1x, you're hired at a casino where you are asked to evaluate the following proposals for encoding the suit of a card (one of "heart", "diamond", "club", or "spade") to determine the number of bit errors that could be detected when using that encoding.

Proposal 1: assign a unique 2-bit code to each choice:

```
heart: 00   diamond: 01   club: 10   spade: 11
```

Number of bit errors that can be detected?: 0 ✔ Answer: 0

Explanation
The minimimum Hamming distance between code words is 1, so no error detection is possible.

Proposal 2: add some more bits to Proposal 1

```
heart: 000   diamond: 001   club: 010   spade: 011
```

Number of bit errors that can be detected?: 0 ✔ Answer: 0

Explanation
The minimimum Hamming distance between code words is still only 1, so no error detection is possible.

Proposal 3: add odd parity to Proposal 1

```
heart: 001  diamond: 010   club: 100   spade: 111
```

Number of bit errors that can be detected?: 1 ✔ Answer: 1

Explanation
Parity has increased the minimimum Hamming distance between code words from 1 to 2, so we can detect up to 1 bit error.

Proposal 4: add even parity to Proposal 3!

```
heart: 0011  diamond: 0101   club: 1001   spade: 1111
```

Number of bit errors that can be detected?: 1 ✔ Answer: 1

Explanation
The minimimum Hamming distance between code words is still only 2, so we can detect up to 1 bit error. Once we've added a parity bit to a code word (so that the number of 1-bits is now either odd or even, depending on the parity scheme chosen), any additional parity bits that get added will be the same for all code words and hence not increase the Hamming distance.

Proposal 5: send the Proposal 1 codes in triplicate

Calculator

```
heart: 00 00 00   diamond: 01 01 01   club: 10 10 10   spade: 11 11 11
```

Number of bit errors that can be detected?: 2 ✔ Answer: 2

Explanation
The minimimum Hamming distance between code words is now 3, so we can detect up to 2 bit errors.

Enviar

Answers are displayed within the problem

Discussion

Topic: 1. Basics of Information / LE1.7

Ocultar discussão

Adicionar publicação

Show all posts by recent activity

How should I calculate min Hamming Distance in Proposal 5 5

Previous Next