MA/CS 109 Third CS Homework Solution

The first three problems are based on the lecture on Error-Detecting and Error-Correcting Codes; the remaining problems are on Cryptography. Please refer to the lecture slides on the class web site before starting the problems.

Problem 1 This problem concerns the "even parity" method for error detection.

(a) Encode the following messages using the even parity code: 1011110, 110110

(b) State whether the following messages, received in the even-parity code (i.e., the last bit is the parity bit), would be reported as errors or not: 10111, 1101110.

(c) As we remarked, the even-parity scheme does not report ALL errors. Supposing that errors, as we assumed, are independent and the probability of a single-bit error is 1/1,000,000 or 10^{-6}, two bit errors is 10^{-12}, 3 bit errors is 10^{-18}, and so on. Suppose you receive a message "1010" in which the last bit is the parity bit, and which, as you see, checks out as a correct message. What is the probability that the message is in fact incorrect (i.e., had errors during transmission)?

Solution

(a) 1011110 1 110110 0

(b) 10111 - correct 1101110 - error

(c) Here are the probabilities of various numbers of errors, according to our assumptions:

# bit errors	Probability
1	10^{-6}
2	10^{-12}
3	10^{-18}
4	10^{-24}

The code can detect any odd number of errors, but will fail to detect an even number of errors. Since the message has only 4 bits, the only way it could be incorrect is it if had 2 or 4 errors, thus $10^{-12} + 10^{-24}$.

Problem 2 This problem concerns the error-correcting code from lecture.

(a) Encode the following four-bit message using the ECC: 1101.
(b) Suppose you receive the following (corrupted) message which has been encoding using the ECC, what is the corrected message? 1010101.
(c) Supposing you wanted to send a message consisting of 75 bits, using a version of the ECC, but with additional parity bits. How many parity bits would you need for 75 message bits?

Solution:

(a) 1101100

(b) 1000101

(c) You would need 7 bits, since then you can encode $2^7 = 128$ possibilities, or 120 message bits plus 7 parity bits.

Problem 3

The ECC we examined can correct any one-bit error, and hence can also detect one-bit errors. But what happens with more than one error? The code can't correct more than one bit error, but can it detect more than one bit error? Consider what happens when the correct message 0011 100 has various multiple bit errors during transmission. Try the ECC algorithm on these erroneous messages:

1111 100	(2 errors)	Solution: finds and corrects 1 error: 1101 100
1000 100	(3 errors)	Solution: finds and corrects 1 error: 1000 101
1011 011	(4 errors)	Solution: finds and corrects 1 error: 1011 001
1100 010	(6 errors)	Solution: finds and corrects 1 error: 1100 011

What happens? Is it possible to detect multiple bit errors or does this get confused with something else? Describe what you think is happening.

Solution: Any bit pattern in the ECC that is not correct will be interpreted as a one-bit error that can be corrected. Therefore if any error is detected, it is assumed it is a one-bit error and corrects it; it can not tell if more than one bit error occurred. "When your only tool is a hammer, everything looks like a nail"!

Problem 4 (Cryptography)

The following message has been encoded with a Caesar cipher. Your task is to decode it without having to consider all 25 different rotations around the alphabet circle. Hint: what is the most common letter in this message? What is the most common letter in English? (The spaces are not part of the message, but will help with readability once you decode it.)

KP VJG DGIKPPKPI IQF ETGCVGF VJG JGCXGPU CPF VJG GCTVJ

Solution:

The most common letter was G, so if this was E, it would be a rotation by 2 places, which gives: "IN THE BEGINNING GOD CREATED THE HEAVENS AND THE EARTH."

Problem 5 (Cryptography)

Suppose you have a block cipher consisting of the rotations (2, 5, 4); to encode you move around the alphabet circle clockwise, and to decode you move around counter-clockwise.

(a) Encode the message HONEYBADGER

(b) Decode the message FTRVHETJ

(c) Supposing your idea of a secure message is one in which you will expect your adversary to check at least million different possible cipher-translations of your ciphertext, how long would your block cipher have to be? [Hint: if it were of length 1, your adversary would have to check 26 cipher-translations, one for each rotation, of length 2, he would have to check 26^2, one for each pair of rotations, etc.]

Solution:

(a) JTRGDFCIKGW

(b) DONTCARE

(c) The powers of 26 are

$26^1 = 26$

$26^2 = 626$

26^3 = 17576

26^4 = 456,976

26^5 = 1,188,1376

Therefore, you would need a block of 5 rotations to force an adversary to consider at least a million different cipher-translations.