CS 70 Discrete Mathematics and Probability Theory
Fall 2013 Vazirani Error-correction Practice

1. **Error-correcting codes**

   In this question we will go through an example of error-correcting codes. Since we will do this by hand, the message we will send is going to be short, consisting of $n = 3$ numbers, each modulo 5, and the number of errors will be $k = 1$.

   (a) First, construct the message. Let $a_0 = 3$, $a_1 = 4$, and $a_2 = 2$; use the polynomial interpolation formula to construct a polynomial $P(x)$ of degree 2 (remember that all arithmetic is mod 5) so that $P(0) = a_0$, $P(1) = a_1$, and $P(2) = a_2$; then extend the message to length $n + 2k$ by adding $P(3)$ and $P(4)$. What is the polynomial $P(x)$ and what is the message that is sent?

   (b) Suppose the message is corrupted by changing a_0 to 0. Use the Berlekamp-Welsh method to detect the location of the error and to reconstruct the original message $a_0a_1a_2$. Show clearly all your work.

2. **Error-Detecting Codes**

   In the realm of error-correcting codes, we usually want to recover the original message if we detect any errors, and we want to provide a guarantee of being able to do this even if there are k malicious errors. Suppose that instead we are satisfied with detecting whether there is any error at all and do not care about the original message if we detect any errors. In class you saw that for recovering from at most k malicious errors when transmitting a message of length n you need to extend your message by $2k$ symbols and send a message of length $n + 2k$. But since we don't require recovering the original message, it is conceivable that we might need less symbols.

   (a) Formally suppose that we have a message consisting of n symbols that we want to transmit. We want to be able to detect whether there is any error if we are guaranteed that there can be at most k malicious errors. How should we extend our message (i.e. by how many symbols should we extend, and how should we get those symbols) in order to be able to detect whether our message has been corrupted on its way? You may assume that we work in $GF(p)$ for a very large prime number p.

   (b) If you were to detect non-malicious errors (i.e. random perturbations of some symbols), how would you do this using just one additional symbol? Obviously you won't be able to guarantee detection, but provide a reason why you think most cases of non-malicious errors will be detected by your algorithm.

3. **Possible Messages**

   Suppose Alice wants to transmit to Bob a polynomial P of degree ≤ 1 over $GF(5)$. She sends packets indicating the values of $P(0), P(1), P(2)$, and $P(3)$ so that Bob will be able to recover P even if two packets are dropped. However, a disaster happens and three packets are dropped: Bob only receives a packet indicating that $P(2) = 3$. Help Bob find a list of all the polynomials that P could have been given this information. (Note that Bob already knowns that the polynomial has degree ≤ 1 and is over $GF(5)$.)

4. **Magic!**

In this problem we will investigate what happens when in error-correcting codes there are fewer errors than the decoding algorithm is able to handle. For the entire problem we are working in $GF(7)$.

Assume that we wish to transfer a message of length 2 which we denote by (m_1, m_2). Each m_i is a member of $GF(7)$. We also wish to be able to correct up to $k = 2$ errors. Using the error-correcting codes we learned in class, we have to first find a polynomial $P(x)$ of degree at most 1 such that $P(1) = m_1$ and $P(2) = m_2$. Then we have to extend the message we send by $2k$ symbols. i.e. we will send $P(1), P(2), P(3), P(4), P(5), P(6)$ to the recipient.

(a) Consider an example where $(m_1, m_2) = (4, 2)$. What are the six symbols that are transmitted?

(b) Now assume that you have received these numbers: $5, 3, 4, 0, 3, 6$. i.e. if there were no errors then we would have $P(1) = 5, P(2) = 3, P(3) = 4, P(4) = 0, P(5) = 3, P(6) = 6$. Now, write down the linear equations that help decode error-correcting codes.

(c) In this part try to solve the linear equations you got in the previous section. You should observe that there are multiple solutions to these equations. Pick two different solutions and for each one write down the error-locating polynomial $E(x)$ and the polynomial $Q(x)$. In each of the two solutions divide $Q(x)$ by $E(x)$ to get the original polynomial. Do you get the same polynomial in both cases?

(d) Factorize $E(x)$ in each one of the two solutions you got to get its roots. Do they have a common root? What does that tell you about the position of errors in the transmitted message?

5. **Check Digits**

In this problem, we'll look at two real-world applications of check-digits.

In the first part, we'll look at International Standard Book Numbers (ISBNs). These are 10-digit codes $(d_1 d_2 \ldots d_{10})$ which are assigned by the publisher. These 10 digits contain information about the language, the publisher, and the number assigned to the book by the publisher. Additionally, the last digit d_{10} is a "check digit" selected so that $\sum_{i=1}^{10} i \cdot d_i \equiv 0 \mod 11$. (*Note that the letter X is used to represent the number 10 in the check digit.*)

(a) Suppose you have very worn copy of the (recommended) textbook for this class. You want to list it for sale online but you can only read the first nine digits: 0-07-288008-? (the dashes are only there for readability). What is the last digit? Please show your work, even if you actually have a copy of the textbook.

(b) Wikipedia says that you can determine the check digit by computing $\sum_{i=1}^{9} i \cdot d_i \mod 11$. Show that Wikipedia's description is equivalent to the above description.

(c) Prove that changing any single digit of the ISBN will render the ISBN invalid. That is, the check digit allows you to *detect* a single-digit substitution error.

(d) Does the check digit allow you to detect all two-digit errors (i.e., all errors where a pair of digits, not necessarily adjacent, are entered incorrectly)?

(e) Can you *switch* any two digits in an ISBN and still have it be a valid ISBN? For example, could 012345678X and 015342678X both be valid ISBNs?

(f) Now we'll look at another checksum formula: the Luhn formula (also known as the Luhn algorithm). This formula is used to verify the validity of credit card numbers. You can read more about it and see an example at `http://en.wikipedia.org/wiki/Luhn_algorithm` The algorithm is as follows:

i. Double each even-positioned digit, when counting from *right to left*.

ii. Determine the sum of the digits from each of the products in step (a).

iii. Sum the numbers from step (b). Find the sum of the unaffected digits (odd-positioned digits) in the original number. Combine these sums.

iv. Verify the account number by determining if the sum from step (c) is equivalent to 0 mod 10.

For clarification, an example from Wikipedia is shown below. In this example, $x = 3$ is the check digit.

Using the Luhn algorithm, determine the check digit x for the following number: $601143871005123x$.

(g) Can this algorithm detect if any two digits are switched? If not, which will it miss and why? *(Hint: you may look on Wikipedia to get started but explain the answer in your own words.)*