CS 70 Discrete Mathematics and Probability Theory
Spring 2016 Rao and Walrand Discussion 6A

1. **Polynomial intersections** Find (and prove) an upper-bound on the number of times two distinct degree d polynomials can intersect. What if the polynomials' degrees differ?

2. **Lagrange or Linear System**

   In this exercise we will try to find out a polynomial $P(x)$ of degree at most 2 with coefficients in $0,\dots,4$ such that $P(1) = 2 \pmod 5$, $P(2) = 4 \pmod 5$, and $P(3) = 3 \pmod 5$.

   (a) Find out the polynomials $\Delta_i(x)$ for $i \in \{1,2,3\}$.

   (b) Combine Δ_i's with the right coefficients to find the polynomial $P(x)$.

   (c) Now we will try a different approach. Write the polynomial $P(x)$ as $c_0 + c_1x + c_2x^2$. Treating c_i's as variables what do the equations $P(1) = 2 \pmod 5$, $P(2) = 4 \pmod 5$, and $P(3) = 3 \pmod 5$ tell about c_i's?

   (d) Solve the system of equations you get from the last part to solve for c_i's. What is the resulting polynomial $P(x)$?

3. **Hamming codes** Here is an example of a simple single-error correcting code that works without having to invoke polynomials and just deals with bits. In real life, these kinds of codes are often used in conjunction with Reed Solomon codes. In this problem we will see how to send 4 bits (each either 0 or 1) using Hamming codes. This will be accomplished by using 3 parity bits (check bits) so that 1 error in the message can be detected and corrected. To encode a 4-bit message m, we will multiply it by a code-generator matrix G:

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

   (a) Show how to encode the message $m = 1101$. What is the resulting 7-bit transmitted codeword x?

   (b) To detect potential errors, we can use a parity check matrix H:

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

   What do you notice about the form of H? If you multiply H by x, what do you get? Will this be true for any message m?

   (c) Suppose that there was an error (flipped bit) in the 6th bit of x, yielding x'. To find out where it occurred, we can multiply H by x'. What is the result base 10? How can you correct x?

   (d) Now we wish to recover the original message m. To do this, we can multiply the corrected x by a recovery matrix R that pulls out the 4 original bits:

$$R = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Do you get m back?

(e) Do you think that a code like this could exist that could fix a single binary error in a 4-bit message without using at least 7 bits for the codeword? Why or why not?

4. **Secret Sharing**

(a) Suppose that the teaching staff of a course involves three professors and two TAs. The solutions to the next homework are encrypted by an encryption key shared by all five. The three professors together should be able to access the solutions, or any one TA with one professor, or both TAs. Suggest a secret-sharing scheme that achieves this. (*Hint:* Try weights.)

(b) Suppose now that the class is taught by three professors, each with *her* own two TAs. Any two professors can access the data, as long as one TA from each is also present. Now what?

5. **Why work with primes?**

In class, you learned about erasure codes and error correcting codes, and prime numbers played a central role in both kinds of codes – since all calculations were supposed to be done modulo a *prime number*. In this problem, we will see why this is a crucial requirement, and explore what happens if this requirement is relaxed in a naive manner.

For this problem, assume that Alice wants to send n packets to Bob, across an "erasure channel" modeled as discussed in class. Let us say all calculations are done modulo $N = 12$ (note that this is *not* a prime number).

As discussed in class, let us say Alice sends $n+1$ packets to Bob, and Bob receives at least n of these packets intact. That is, the channel can erase at most 1 packet, and if it does so, Bob gets to know which packet was erased (although he does not know the contents of the erased packet).

a) Suppose $n = 1$. That is, Alice wants to send only 1 packet to Bob (plus one redundant packet to compensate for erasure). Would the scheme discussed in class work with $N = 12$? What are all the possible 2-packet lists that Alice could transmit? In each case, would Bob be able to recover Alice's message in spite of a possible erasure? Would Alice or Bob face any problems because they are doing their calculations modulo 12?

b) Now suppose $n = 2$. That is, Alice now wants to send 2 packets to Bob (plus one redundant packet to compensate for erasure). Now, would there be any problems because $N = 12$?

c) Now let $n = 3$ (3 packets plus one additional packet to compensate for erasure). Assume that Alice wants to encode messages into "systematic" codewords (with the first few evaluations of the polynomial being the message itself). Prove that Alice can no longer send arbitrary messages of her liking to Bob, by showing that it would be impossible for Alice to send the message $(11, 6, 2)$. Find 2 other examples of messages that Alice cannot send to Bob.

d) Why does Lagrange interpolation fail when Alice tries to send the messages in part (c) above? Does Lagrange interpolation fail for any message in part (a) or part (b)? Why or why not?

e) Suppose Alice chooses a value for n, and restricts herself to only the messages that are possible to send in modulo 12 arithmetic. For any such n, and any such message, would Bob be able to recover Alice's message in spite of a possible packet erasure? If so, how? If not, provide a counter-example.