SIMU 2001 – Seminar in Coding Theory
Discussions
*Draft – May 24, 2001*

*Goals*

The discussion meeting each day of the seminar will be devoted to preliminary work on an assigned list of problems in groups of 3. Each group will probably need to continue work on the assignment in the afternoon "free period" when they are not in lab, and also during the evening. The group's solutions will be due at the next meeting of the seminar.

The goals are to practice using the concepts we have discussed in the lectures, to work through a large collection of examples, to collect data for later use, and to practice working on mathematics *collaboratively*.

A few words about this way of working are probably in order.

In these discussion meetings, we will be aiming for *collaborative learning* – that is, for an integrated group effort in analyzing and attacking the discussion questions. The ideal is for everyone in each of the groups to be fully involved in the process. The idea is that, by actively participating through talking about the ideas yourself in your own words, you can come to a better first understanding of what is going on than if you simply listen to someone else talk about it.

However, it must be said that to get the most out of this kind of work, you may have to adjust some of your preconceptions. In particular:

- This is *not a competition* in any sense. You and your fellow group members are working as a team, and the goal is to have everyone understand what the group does fully.
- At different times, it is inevitable that different people within the group will have a more complete grasp of what you are working on and others will have a less complete grasp. Dealing with this a group setting is excellent preparation for real work in a team; it also offers opportunities for significant educational experiences:
  a) If you feel totally "clueless" at some point, you need to feel free to ask questions and even pester your fellow group members until the point has been worked out to your full satisfaction. (Don't forget, the others may be jumping to unwarranted conclusions, and your questions may save the group from pursuing an erroneous train of thought!)
  b) On the other hand, when you think you do see something, you may need to explain it carefully to others. (Don't forget, the *absolutely best* way to make sure you really understand something is to try to explain it to someone else(!) If you are skipping over an important point in your thinking, it can become very apparent when you set out to explain your ideas to a team member.)

  In short, everyone has important things to contribute, and everyone will contribute in different ways at different times.

*Week 1*

*Day 1 – Basic Notions of Coding Theory*

We have introduced the basic language of binary block codes (codes consisting of collections of binary words of some fixed length $n$); in particular, we have defined the *Hamming distance* between two words $v, w$ to be $d(v, w) =$ the number of positions in which the words differ.

A) Show that the Hamming distance has the following properties.
   1) For all $v, w$, $d(v, w) \geq 0$ and $d(v, w) = 0$ only when $v = w$.
   2) For all $v, w$, $d(v, w) = d(w, v)$.
   3) For all $u, v, w$, $d(u, w) \leq d(u, v) + d(v, w)$. (This is a triangle inequality.)

B) Determine the *minimum distance* of the following codes.
   1) $C = \{1010, 1110, 0011\}$
   2) $C = \{000000, 101010, 010101, 111111\}$
   3) The code obtained by adding a parity check digit to $K^n$, any $n$.

C) Suppose a binary symmetric channel with reliability $p = .999 = 1 - 10^{-4}$ is used for communication of words of length $n = 11$ at a rate of $10^5$ binary digits per second.
   1) With no error control, how often could we expect a word to be transmitted incorrectly (how many incorrect words per second)? Indicate whether your answer is exact or an approximation and explain.
   2) Now, suppose that a parity-check digit is added to all words (making them length $n = 12$), so that a single bit error can be detected. How often could be we expect a word to be transmitted incorrectly, but *not detected* (how many per second)? Indicate whether your answer is exact or an approximation and explain.
   3) We will see in a few days that it is possible to encode words of length 11 by adding *four additional parity check digits* to each one (making them length $n = 15$) in such a way that the resulting code has minimum distance 3. Hence a maximum likelihood decoder can correct any single bit error in a received word. For this coding scheme, no other errors can be corrected, though, and if any two or more bit errors occur in a received word, it will be decoded incorrectly. Assuming we have a decoding algorithm for this code (we'll see how to construct one), how often could we expect a word to be received but *decoded incorrectly* (how many per second)?
   4) Discuss the tradeoffs in efficiency and reliability of communication between scenarios 1,2,3.

D) Error control in the form of check digits is also often used for messages over non-binary alphabets. A possibly familiar case is the ISBN numbers used by publishers to identify books. For instance the ISBN number of the HHLLPRW text we are using is 0824704657 (see the barcode on the back cover – the dashes are omitted since they are not part of the actual code). All ISBN's have the form $d_1 d_2 \cdots d_9; d_{10}$

where the first 9 digits are ordinary decimal digits 0 - 9 and are the actual identifier for the book. The last digit $d_{10}$ is a check digit that is added to make the number $d_1 + 2d_2 + 3d_3 + \cdots + 9d_9 + 10d_{10}$ evenly divisible by 11. For example, the ISBN for our text has:

$$1 \cdot 0 + 2 \cdot 8 + 3 \cdot 2 + 4 \cdot 4 + 5 \cdot 7 + 6 \cdot 0 + 7 \cdot 4 + 8 \cdot 6 + 9 \cdot 5 + 10 \cdot 7 = 264 = 24 \cdot 11.$$

(In cases where the check digit must be $d_{10} = 10$, a symbol $X$ is used instead to keep the total length to 10 symbols.)

1) Explain why, given any nine digits $d_1, \ldots, d_9$, there is always an integer $d_{10} \in \{0, 1, 2 \ldots, 10\}$ such that $d_1 + 2d_2 + 3d_3 + \cdots + 9d_9 + 10d_{10}$ is divisible by 11. Also show the check digit serves to *detect* any error in a single digit of an ISBN.
2) Probably the most common error involving more than one digit made by humans copying codes like ISBN's is to *transpose* or interchange two symbols. Show that the ISBN code can also be used to *detect any single transposition error.* Also show by example that correcting a transposition error is not always possible.

*Challenge Question*

In addition to the situation we discussed in class where an error in a received word changes a 0 to a 1 or a 1 to a 0, there are also instances in which a digit in a received word is simply not recognizable as either a 0 or a 1. In this case we say that an *erasure* has occurred in that location. Show that if a code has minimum distance $d$ and any combination of $t$ errors (not erasures) and $e$ erasures with $2t + e < d$ errors occurs in a received word, then there is a unique closest codeword. Hence maximum likelihood decoding will correct the errors and erasures. Which are *easier to correct* – errors or erasures? Why?

*Day 2 – Linear Codes*

A linear code $C$ is a vector subspace of $K^n$ for some $n$ (as in our text, $K$ is the binary field $\{0, 1\}$ with addition and multiplication modulo 2). One of the first advantages of linear codes is that they let us bring most of the familiar tools of linear algebra – vectors, matrices, linear independence, bases for subspaces and the notion of dimension – to bear to study questions about codes. We will always use the following notation for linear codes:

$$\begin{cases} n & \text{is the block length} \\ k & \text{is the dimension (information bits per codeword)} \\ d & \text{is the minimum distance} \\ G & \text{is a generator matrix (rows form a basis of } C) \\ H & \text{is a parity check matrix} - GH = 0 \end{cases}$$

A) For each of the following sets, list the elements of the linear code $C = \langle S \rangle$, give the parameters $(n, k, d)$ of $C$, give a generator matrix, a parity check matrix, and find a basis of the dual linear code $C^\perp$.

1) $S = \{0101, 1010, 1100\}$
2) $S = \{11000, 01111, 11110, 01001\}$
3) $S = \{10001101, 01001000, 11001111, 01100101\}$

B) Prove Theorem 2.3.19 from the text. (Hint: construct the basis one vector at a time.)

C) Exercise 2.6.12 from the text.

*Challenge Question:* The "$(u|u+v)$ construction" (this, with other variants of the same idea, is sometimes used to produce longer codes from known shorter codes, with good control over the resulting minimum distance). Let $C$ be a linear $(n, k, d)$ code and let $D$ be a linear $(m, l, d')$ code. Show that the set $E$ of words (of length $m + n$) of the form $(u|u + v)$ where $u \in C$ and $v \in D$ forms a linear code. What can you say about the dimension and the minimum distance of $E$? (Don't expect a simple nice formula for the minimum distance – it depends on what words are in $C \cap D$. How does this work? Try to find a way to express the weight of a word $u + v$ in terms of the weights of $u$ and $v$ and the "overlaps" between positions of 1's in $u$ and $v$.) code.

*Day 3 – Syndrome Decoding*

Another big advantage of linear codes (as compared with general codes) is that they admit a simpler decoding method, called *syndrome decoding*. This is based on the fact that for linear codes, the same error correction/processing should be performed for all received words with the same "syndrome" $s = wH$. If $w = v + e$ where $v$ is a codeword and $e$ is the error vector, then $s = wH = vH + eH = 0 + eH = eH$. *So the syndrome depends only on the error, not on the code word that was transmitted.* In addition, the possible values of the syndrome are in 1-1 correspondence with the cosets of the code in $K^n$, and the likeliest error is the minimum weight element of the coset ("coset leader").

A) Exercise 2.10.7 from the text.
B) Prove Theorem 2.10.3 from the text.
C) Prove Theorem 2.11.4 from the text.
D) Exercise 2.11.21 from the text.

*Day 4 – Bounds on the Parameters of Codes, Hamming Codes*

"Good" codes are ones for which the number of codewords (hence $k$) is big, but the minimum distance is also big, so that many errors can be detected or corrected. Unfortunately, it is not easy to achieve these goals simultaneously – these requirements are mutually contradictory to some extent. So the search for good codes is a "balancing act". We have introduced several bounds on the parameters of codes:

*Hamming (sphere packing) Bound*: If $C$ is a code of length $n$ and minimum distance $d = 2t + 1$ or $2t + 2$, then

$$|C| \le \frac{2^n}{\sum_{i=0}^{t} \binom{n}{i}}$$

*Singleton Bound*: For any $(n, k, d)$ linear code, $d \leq n - k + 1$ (or $k \leq n - d + 1$).
*Gilbert-Varshamov Bound*: There exists a linear $(n, k, d)$ code if $2^{n-k} > \sum_{i=0}^{d-2} \binom{n-1}{i}$.

Today, we will consider some of the implications of these theoretical statements and study one particularly famous family of codes, the *Hamming codes.*

A) Exercises 3.1.5 and 3.1.18 from the text.
B) Exercise 3.1.19 from the text.
C) Exercise 3.3.5 from the text.
D) Exercise 3.3.7 from the text. (See §2.8.)
E) Exercise 3.4.5 from the text.

*Challenge Question*

The form of the Gilbert-Varshamov inequality given in Theorem 3.1.13 in our text was proved by considering how parity checks could be added one by one to produce a parity check matrix for a code of dimension $k$. An alternate approach is to consider adding basis vectors for a code one by one. Show that if that second approach is taken, then an alternate inequality as follows can be derived. A linear $(n, k, d)$ code exists as long as

$$2^{n-k+1} > \sum_{i=0}^{d-1} \binom{n}{i}.$$

By analogy with Corollary 3.1.14 in the text, if $n, d$ ($\neq 1$) are given, then there exists a code with block length $n$, distance at least $d$, and

$$|C| \geq \frac{2^n}{\sum_{i=0}^{d-1} \binom{n}{i}}$$

(Note the differences between this and the form given in the text!) Prove these statements. Also – which of these forms is a "stronger" result (e.g. for a fixed $n$ and $d$, does one of them always give a larger lower bound for the dimensions $k$ of the best possible code? Or does the answer depend on $n, d$?) Experiment!

*Week 2*

*Day 1 – The Golay Codes*

Today, we will study properties of two famous linear codes with parameters $(24, 12, 8)$ and $(23, 12, 7)$, called the extended Golay code and the Golay code respectively. These are amazing intricate codes with interesting features. In addition, they have been used in practice – they were used in the Voyager spacecraft program in the 1980's to transmit photographs of Jupiter and Saturn back to Earth. (The long distances and large amount of interference from solar and other radiation involved in these transmissions made a high error-correction capability vital.) Our textbook introduces a generator matrix for the $(24, 12, 8)$ code with no indication where it comes from (see page 77). For the first group of problems below, you should use that generator matrix and deduce the results from that representation of the extended Golay code. The Challenge Problem for this week, and a portion of your work in the lab, will give another construction of this code, indicating one way intricate codes like this can sometimes be constructed from simpler codes.

A) Exercises 3.5.2 and 3.5.3 from the text.
B) A general operation called *puncturing* takes a given code $C$ and produces a second code $C'$ with a smaller $n$ as follows – we simply take all the codewords from $C$ and delete some particular entry (or more generally some fixed collection of entries) from all of them. If the original code $C$ is a linear code with parameters $(n, k, d)$, what can you say about the parameters $(n', k', d')$ of the punctured code $C'$ (puncturing in one particular location)?
C) The Golay code $C_{23}$ is obtained by puncturing the extended Golay code in the last position (see §3.7 in the text). This is a linear $(23, 12, 7)$ code. Given this, verify that $C_{23}$ is also a perfect code.
D) Exercises 3.7.7-3.7.9 from the text.

*Challenge Problem for the Week*

It is a famous theorem of van Lint and Tietäväinen that *the only perfect binary codes* are the Golay code and codes with the same parameters $n, k, d$ as a Hamming code: $n = 2^r - 1$, with $r \geq 2$, $k = 2^r - 1 - r$, $d = 3$. Moreover, as was shown by Pless, every binary code with the same parameters $(n, k, d) = (24, 12, 8)$ as the extended Golay code is *equivalent* to the extended Golay code $C_{24}$ in the following sense: *Two binary block codes of length n are said to be equivalent if there is some fixed permutation of the entries of $K^n$ that takes the first code to the second.* In this problem, you will work through a second construction of the Golay $(24, 12, 8)$ code. In the lab this week, you will see other facts showing that this construction actually gives a code that is equivalent to the code given by the generator matrix from page 77 of the text – something that is not at all obvious(!)

Let $H$ be the $(7, 4, 3)$ Hamming code with generator matrix:

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

and let $K$ be the code obtained from $H$ by *reversing the order of the bits* in each codeword. Let $H'$ and $K'$ be the $(8, 4, 4)$ extended codes obtained from $H$ and $K$ by adding a parity check bit to each codeword at the right end (making the number of 1's even in each case). Consider the code $C$ obtained by forming all words (length 24) of the form $(a+x|b+x|a+b+x)$ where $a, b \in H'$ and $x \in K'$. (Note that this can be seen as a "souped up" version of the "$(u|u+v)$ construction" we saw last week.) In the lab you will see by computing the codewords directly that all codewords of $C$ have weight either $0, 8, 12, 16$, or $24$. This can also be proved as follows

1) Show in general that $wt(u+v) = wt(u)+wt(v)-2j(u,v)$ where $j(u,v)$ is the weight of the component-wise product vector $u*v$ (the $i$th component of $u*v$ is $u_iv_i$). Similarly, show that

$$wt(u+v+w) = wt(u) + wt(v) + wt(w) - 2j(u,v) - 2j(u,w) - 2j(v,w) + 4j(u,v,w),$$

where similarly $j(u,v,w)$ is the weight of the triple componentwise product.
2) Show that every codeword of $C$ has weight divisible by 4.
3) Show that no codeword of $C$ has weight equal to 4, so the minimum distance is $d = 8$.
4) Exercise 4.4.10 – Show that the Golay $(23, 12, 7)$ code is *equivalent* to a cyclic code. "Equivalence" here means that there is a fixed permutation of the 23 entries in the Golay codewords that takes the Golay code to a cyclic code.
5) One proof of the uniqueness of the Golay code up to equivalence is based on the following observation. Show that for every 5-element subset of $\{1, 2, 3, \ldots, 24\}$, there is *exactly one* weight-8 codeword of the code $C$ containing 1's in the entries corresponding to the elements in the subset. (The codewords of weight 8 in $C$ form a combinatorial structure called a Steiner system $S(5, 8, 24)$. There is only one such object, up to permutations of the entries in the vectors of length 24!)

*Day 2 – Begin Cyclic Codes*

Many of the codes that have been used in applications such as the CD audio system are linear codes with "extra structure" that facilitates encoding and decoding. One important class is the class of *cyclic* codes. We will largely focus on these for the remainder of the Coding Theory Seminar. The most basic way to say what a cyclic code is to say that it is a linear code where the set of codewords is *closed under cyclic permutations.* We have seen in class that the algebra of polynomials is very useful to describe these codes. This is true because if we make the digits from a codeword of length $n$ the coefficients of a polynomial $p(x)$ of degree $\leq n-1$, then a cyclic permutation is achieved by taking the remainder of $xp(x)$ on division by $x^n + 1$. Today, we want to begin by practicing on some polynomial arithmetic with coefficients mod 2, then apply this to study some cyclic codes.

A) Exercises 4.1.19 - 4.1.21 from the text.
B) Exercise 4.2.20 from the text. (Use the polynomial forms of the codewords and the Euclidean Algorithm for the gcd. See Appendix A and the class notes.)
C) Exercise 4.2.22 from the text. (Note: The code $\langle S \rangle$ is cyclic in each case.)

*Day 3 – More on Cyclic Codes*

Continuing our study of cyclic codes, we want to consider how to find generator polynomials for all cyclic codes. The most important fact here is that the generator polynomial for a binary cyclic code of length $n$ must be a divisor of the polynomial $x^n+1$. This shows that we must consider the irreducible factorization of $x^n + 1$ in the ring of polynomials with coefficients in the binary field $K$, written $K[x]$.

(We will always assume $p = 2$ and $n$ is odd for the next description of binary cyclic codes.) Cyclic codes are also described by *idempotent polynomials* – polynomials $p(x)$ such that $(p(x))^2$ is congruent to $p(x)$ mod $x^n + 1$. (That is the remainder of $(p(x))^2$ is equal to $p(x)$ when we divide by $x^n + 1$.) The basic idempotent polynomials are in 1-1 correspondence with the *cyclotomic cosets* in $\{0, 1, 2, \ldots, n\}$. The cyclotomic coset $C_i$ is the set

$$C_i = \{2^j i \pmod{n} : 0 \le j \le r - 1\}$$

where $2^r \equiv 1 \pmod{n}$ (this is where the hypothesis that $n$ is odd is used). For instance with $n = 15$, we have cyclotomic cosets:

$$\begin{aligned}
&C_0 = \{0\}\\
&C_1 = \{1, 2, 4, 8\} = C_2 = C_4 = C_8\\
&C_3 = \{3, 6, 12, 9\} = C_6 = C_{12} = C_9\\
&C_5 = \{5, 10\} = C_{10}\\
&C_7 = \{7, 14, 13, 11\} = C_{14} = C_{13} = C_{11}
\end{aligned}$$

and corresponding basic idempotents are

$$p_0(x) = 1, p_1(x) = x + x^2 + x^4 + x^8, p_3(x) = x^3 + x^6 + x^{12} + x^9, \ etc.$$

The other idempotents are obtained as linear combinations:

$$a_0 p_0(x) + a_1 p_1(x) + a_3 p_3(x) + a_5 p_5(x) + a_7 p_7(x)$$

where $a_i \in K$. A generator for a binary cyclic code will be obtained by taking the gcd of each idempotent and $x^n + 1$.

A) Exercises 4.3.4 - 4.3.6 from the text.
B) Exercise 4.4.15 from the text.
C) Let $g(x)$ be the generator polynomial for a binary cyclic code $C$ of length $n$, and let $1 + x^n = g(x)h(x)$. Show that the dual code $C^\perp$ is also cyclic. What does a generator polynomial for $C^\perp$ look like? Try some examples using the "data" generated in problem B, and see if you can find a pattern. (See Theorem 4.5.2 in the text for a formal statement.)

*Day 4 – Finite Fields*

To prepare for the BCH and Reed-Solomon codes we will study next week, we need next to introduce other finite fields besides the binary field $K$. All the finite fields we consider will contain $K$, however. The Reed-Solomon codes will be cyclic codes over alphabets equal to one of these other fields (just as binary codes are codes over the alphabet $K$).

A) Exercises 5.1.15 - 5.1.18 from the text.
B) Exercises 5.2.7 - 5.2.9 from the text.

C) Show that if $N_k$ is the number of monic irreducible polynomials of degree $k$ in $K[x]$, then

$$\frac{1}{1-2t} = \sum_{n=0}^{\infty} 2^n t^n = \prod_{k=1}^{\infty} \frac{1}{(1-t^k)^{N_k}}$$

Deduce that there are monic irreducibles of all degrees, hence finite fields of every order $2^r$, $r \geq 1$.

*Week 3*

*Day 1 – Hamming codes, reconsidered; BCH codes*

This week, we will study two kinds of cyclic codes for which there exist both extremely encoding and decoding algorithms. Both of these types of codes rely heavily on the properties of the finite fields of order $2^r$ that we constructed last week. First we will reconsider the Hamming codes from Week 1. We will see that a Hamming $(2^r-1, 2^r-1-r, 3)$ code can be constructed as a cyclic code with generator polynomial the minimal polynomial of a primitive element of a field of order $2^r$. As we know, Hamming codes always have $d = 3$, so they can correct just one error in each received word. One of the main reasons for following this train of thought is that we can see how to construct codes with larger $d$ this way in a quite systematic way. The resulting codes are called BCH codes for the people who developed them first (Bose, Chaudhuri, and Hocquenghem).

A) Exercise 5.3.4 from the text.
B) Exercise 5.3.9 from the text.
C) In class, we saw the construction of BCH codes with $n = 2^r - 1$, $k = 2^r - 2r - 1$, and $d = 5$ (so 2 errors in a received word can be corrected). In this problem, you will see how this construction can be generalized to yield BCH codes with "designed distance" as large as we like (provided $n = 2^r - 1$ is sufficiently large, of course). The actual minimum distance will be bigger than or equal to this designed distance. The most direct way to give the definition is via properties of polynomials.

   1) Let $\beta$ be a primitive element of the field of order $2^r$ (constructed using some particular irreducible polynomial of degree $r$). If the parity check matrix of a code contains a block of columns corresponding to the binary expansion of

   $$(1, \beta^t, (\beta^t)^2, \ldots, (\beta^t)^{2^r-2}) \tag{1}$$

   show that the polynomial $c(x)$ corresponding to any codeword has $\beta^t$ as a root: $c(\beta^t) = 0$. Deduce that $\beta^\ell$ for all $\ell$ in the cyclotomic coset $C_t$ mod $n = 2^r - 1$ are also roots of $c(x)$. Hence, the minimal polynomial of $\beta$ divides $c(x)$. Conversely, if $c(\beta^t)$ for all $c(x)$, then we can take a block of columns corresponding to (1) in a parity check matrix for the code.

   We can define a general BCH code as follows. Given any $\delta \geq 1$ let $BCH(2^r, \delta)$ be the code consisting of polynomials $c(x)$ such that

   $$c(\beta) = c(\beta^2) = \cdots = c(\beta^{\delta-1}) = 0$$

Show that $c(x)$ is in $BCH(2^r, \delta)$ if and only if $c(x)$ is divisible by the lcm of the minimal polynomials of $\beta, \beta^2, \ldots, \beta^{\delta-1}$ in $K[x]$. (Of course if $\delta$ is taken to be very large, the only $c(x)$ we will get will be $c(x) = 0$.

2) Let $g(x)$ be the lcm of the minimal polynomials of $\beta, \beta^2, \ldots, \beta^{\delta-1}$ in $K[x]$. (Note the coefficients of $g(x)$ are all either 0 or 1.) Show that $g(x)$ is a generator polynomial for $BCH(2^r, \delta)$. Deduce that the dimension is $k = 2^r - 1 - \deg(g(x))$.
3) Show that the minimum distance of $BCH(2^r, \delta)$ is at least $\delta$. (Hint: Consider columns of a parity check matrix of the form (1) for $t = 1, 2, \ldots, \delta - 1$, and use Vandermonde determinants.

The integer $\delta$ is called the *designed distance* of the BCH code. The actual minimum distance can be *strictly larger* if there is a longer string of consecutive powers of $\beta$ that are roots of $g(x)$. This happens rather frequently because of the fact noted above that if $\beta^t$ is a root of $g(x)$ then so are the $\beta^\ell$ for all $\ell$ in the cyclotomic coset of $t$ mod $2^r - 1$. We will find an explicit example where this happens in the lab.

*Day 2 – Reed Solomon codes*

We have now seen the Reed-Solomon codes as well. These have close connections with BCH codes. However, there are important differences: BCH codes are *always binary codes*, while Reed-Solomon codes are codes over the alphabet $\mathbf{F}_{2^r}$ for some $r$. A BCH code is a *subfield subcode* of a Reed-Solomon code – the subset of the Reed-Solomon code consisting of words with only $0, 1$ as entries. The parameters of a Reed-Solomon code are also much more predictable than those of a BCH code. The codes $RS(2^r, \delta)$ are defined to be the cyclic linear codes over the field $GF(2^r) = \mathbf{F}_{2^r}$ with generator polynomial

$$g(x) = (x + \beta^{m+1})(x + \beta^{m+2}) \cdots (x + \beta^{m+\delta-1}) \tag{2}$$

for some $m$ (almost always we will take $m = 0$, so the zeroes are $\beta, \beta^2, \ldots, \beta^\delta - 1$). We have $n = 2^r - 1$, $k = 2^r - \delta$ (but careful, this is the dimension as a vector space over the field $\mathbf{F}_{2^r}$), $d = \delta$. One important property of the Reed-Solomon codes is that they all reach the Singleton bound from Week 1: $d = n - k + 1$. In other words, they have the maximum possible $d$ for the given $n$ and $k$(!).

An important alternate approach to constructing these Reed-Solomon codes is the following. Let $L_k$ be the vector space of polynomials $f(t)$ of degree $< k$. The Reed-Solomon code $RS(2^r, 2^r - k)$ with $m = 0$ in (2) can also be constructed by considering the image of the evaluation mapping

$$\begin{aligned} ev : L_k &\to \mathbf{F}_{2^r}^{2^r - 1} \\ f &\mapsto (f(1), f(\beta), \ldots, f(\beta^{2^r - 2})) \end{aligned}$$

($\beta$ is a primitive element of $\mathbf{F}_{2^r}$). In the text's notation we are taking $S = \mathbf{F}_{2^r} - \{0\}$ here, the set of all nonzero elements of the field. We can also consider evaluation mappings on other spaces of polynomials.

A) Exercise 6.1.7 from the text.
B) Exercise 6.2.8 from the text.
C) Exercise 6.4.16 from the text.
D) Exercise 6.4.17 from the text.

*Days 3,4 – The Berlekamp-Massey Decoding Algorithm*

One of the reasons that Reed-Solomon codes have been used frequently for real-world applications is that a very efficient decoding method called the Berlekamp-Massey algorithm is known for them. Let $\delta = 2t + 1$ and consider the $RS(2^r, \delta)$ code. Given a received word with a correctable error pattern, the Berlekamp-Massey algorithm produces polynomials $\sigma_R(x)$, $q(x)$, $r(x)$ satisfying the "key equation":

$$\sigma_R(x)s(x) = q(x)x^{2t+1} + r(x)$$

where $s(x)$ is the "syndrome polynomial"

$$s(x) = 1 + s_{m+1}x + \cdots + s_{m+2t}x^{2t}$$

calculated from the received word $w$ by $s_j = w(\beta^j)$. (Recall that the codewords of the Reed-Solomon codes have $\alpha^{m+1}, \ldots, \alpha^{m+2t} = \alpha^{m+\delta-1}$ as roots, so the $s_j$ measure the failure of the received word to be a codeword: $s_j \neq 0$ for some $j$ if and only if some error occurred in transmission.) Also, $\sigma_R$ is the "error locator polynomial" of degree $\leq t$ that can be used to find the error locations, and $\deg(r(x)) \leq t$.

A) Exercise 6.5.4/a,c,e from the text.
B) Exercise 6.5.5 from the text