CS250/EE387: Error Correcting Codes — M. Wootters
Problem Set 2 — Winter 2022
Due: by 11:59pm on Friday February 4, 2022, on Gradescope.

Instructions:

- Please complete all problems in Section 1.
- Try to complete 3 of the problems in Section 2. You are welcome to do more than 3, but please indicate which 3 you want graded.
- No problems in Section 3 are required, but they might be fun to think about (some might be open-ended).
- Problems are labeled with the class number after which you should be able to do them. (This is to aid your time management since all HWs are posted up front).
- Note: Small typos have been fixed in red. Sorry about these, and thanks to those who pointed them out!

Guidelines/rules:

- You are encouraged to work in groups (up to 3-ish); each group should turn in one HW assignment.
- You and your group may collaborate on problems in Sections 1 and 2 with other members of the class; please acknowledge your collaborators. You may consult lecture notes, Essential Coding Theory and other posted readings, but please do not use any other written resources (that is, please do not Google for the answers to the questions). It is fine to use computational resources like Sage or Mathematica if you want to.
- You may collaborate on Section 3 problems with anyone, whether or not they are in the class; please acknowledge your collaborators. You may also use whatever resources you want: Googling, reading research papers, etc, is fine.

Typing up your solutions in LaTeX is encouraged (but I don't type up my lecture notes, so I can't be too strict). Legibility and complete sentences are required.

Section 1

1. (4 pts, Class 4) The finite field $\mathbb{F}_4$ is the set $\{0, 1, \gamma, \gamma^2\}$, with multiplication and addition given below.

$+$	0	1	γ	γ^2
0	0	1	γ	γ^2
1	1	0	γ^2	γ
γ	γ	γ^2	0	1
γ^2	γ^2	γ	1	0

$\times$	0	1	γ	γ^2
0	0	0	0	0
1	0	1	γ	γ^2
γ	0	γ	γ^2	1
γ^2	0	γ^2	1	γ

   Answer the following questions, with a brief justification.

   (a) What are the primitive elements of $\mathbb{F}_4$?
   (b) Write down all the codewords of $RS_4(\{1, \gamma, \gamma^2\}, 3, 2)$ whose first coordinate is 1.
   (c) What are the codewords of $RS_4(\{1, \gamma, \gamma^2\}, 3, 2)$ that are contained in $\{0, 1\}^3$?

2. (4 pts, Class 4) Show that the dual of an MDS code is also an MDS code.

3. (4 pts, Class 6) Let $\mathcal{C} = RS_8(\mathbb{F}_8^*, 7, 5)$. Give a lower bound on $|\mathcal{C} \cap \mathbb{F}_2^7|$. (Not required: is your lower bound tight?)

   (Yes, 0 is a lower bound on this quantity...the problem is asking for a nontrivial lower bound. In particular, there is a bound we proved in lecture that might be relevant here....)

Section 2

(Section 2 problems are worth 10 points each; please do at least 3 of them.)

1. (**t-wise independent sources**, Class 4) A set $S \subseteq \mathbb{F}_q^n$ is said to be a *t-wise independent source* (for some $1 \leq t \leq n$) if given a uniformly random sample $X = (X_1, \ldots, X_n) \in S$, the n random variables $X_1, X_2, \ldots, X_n$ are t-wise independent: that is, any subset of t of these variables are uniform independent random variables over $\mathbb{F}_q$.

   (a) Suppose that $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a linear code of dimension k, so that no coordinates are identically zero. (That is, there is no $i \in [n]$ so that $c_i = 0$ for all $c \in \mathcal{C}$). Show that $\mathcal{C}$ is a 1-wise independent source.

   Hint: It might be useful that if $x \in \mathbb{F}_q^k$ is a random vector, and $y \in \mathbb{F}_q^k$ is any *non-zero* vector, the random variable $\langle x, y \rangle$ is uniformly distributed in $\mathbb{F}_q$. (You can use this fact without proving it, but convince yourself of it first!)

   (b) Show that any linear MDS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ of dimension k is a k-wise independent source.

   (c) Prove that there exists a k-wise independent source $S \subset \mathbb{F}_2^n$ of size $|S| \leq \left(\frac{4n}{\log_2(n)}\right)^k$. (Notice that the field is $\mathbb{F}_2$, not $\mathbb{F}_q$). Conclude that $k(\log_2(n) + 2 - \log\log(n))$ +1 random bits are enough to compute n random bits that are k-wise independent. (Note: the constant "4" in the bound on $|S|$ is somewhat arbitrary, don't worry about matching it exactly).

   Hint 1: From the previous part, you have a k-wise independent source over a large field...you need to somehow turn those large field elements into bits....

   Hint 2: In the context of Hint 1, don't think too hard.

   (d) For any $p \in (0, 1/2)$, we say that n binary random variables $X_1, \ldots, X_n$ are p-biased and t-wise independent if any of the t random variables are independent and $\mathbb{P}\{X_i = 1\} = p$ for every $i \in [n]$. For the rest of this part, let p be a power of $1/2$. Show that any collection of $(t \log_2(1/p))$-wise independent random variables can converted into a collection of (possibly fewer) t-wise independent p-biased random variables. Conclude that one can construct such sources with $t \log_2(1/p)(\log_2(n) + 2 - \log\log(n) + \log\log(1/p))$ uniformly random bits.

   Hint: How would you turn $\log_2(1/p)$ unbiased bits into a single p-biased bit?

   Note: Above, "possibly fewer" is a bit vague – here you should have "fewer" so that you get quantitatively right answer in the "Conclude that..." line. The hint might give a clue as to what "fewer" should mean.

2. (**Special error patterns**, Class 4) Consider the following problem which is related to the error-correction problem for Reed-Solomon codes:

   - INPUT: $c + e$, where $c \in RS_q(\mathbb{F}_q^*, q-1, k)$, and $e \in \{0,1\}^n$ with $\mathrm{wt}(e) \leq n - k$. (Here $n = q - 1$ is the length of the RS code, and as usual we order $\mathbb{F}_q^* = \{\gamma, \gamma^2, \ldots, \gamma^{q-1}\}$ for a primitive root γ).
   - OUTPUT: c and e.

   The difference between this and the problem of decoding RS codes from errors is that (a) the error vector e has only 0's and 1's, but (b) it might have many more errors than half the distance. Suppose

that q is prime. Prove that this problem is solvable (despite the large number of errors) and give a polynomial-time algorithm to solve it.

Hint 1: You can compute $H(c+e) = He$, where H is the parity-check matrix for the RS code.

Hint 2/supplementary resource: You are allowed to consult the Wikipedia page on Newton's identities.

3. (**Streaming algorithms**, Class 4) A *streaming algorithm* is an algorithm that sees its input one symbol at a time, in a streaming fashion, and has limited memory so that it cannot store all of the input. In this problem you'll explore some applications of Reed-Solomon codes to streaming algorithms, and also develop a streaming algorithm for Reed-Solomon codes!

   (a) Suppose that you have black-box access to a polynomial $f \in \mathbb{F}_q[X]$ of degree d, where $d < q/3$. (That is, for a point $\alpha \in \mathbb{F}_q$, you can evaluate $f(\alpha)$ in time $O(1)$.) Give a randomized algorithm which will decide whether or not f is identically zero (that is, if all its coefficients are zero) in time $O(1)$. Your algorithm should be correct with probability at least $2/3$.

   (b) Fix a matrix $A \in \mathbb{F}_q^{m \times n}$, for $m \leq n$. Suppose you have oracle access to A: that is, there is a magic box, M, so that in time $O(1)$, $M(i,j)$ returns $A_{i,j}$. Give a randomized streaming algorithm that takes in an input $y \in \mathbb{F}_q^n$ (in a streaming fashion, so it sees y_1, then y_2, then y_3 and so on until y_n), and outputs its best guess about whether or not $Ay = 0$.

   Your algorithm is allowed to call the oracle M and also it can do an arthmetic operation (addition, multiplication, inverses) over an arbitrary field of size $\text{poly}(q, n, m)$ in time $O(1)$.

   Your algorithm should use space at most $O(\log(q) + \log(m) + \log(n))$ bits, and spend $O(m)$ time per entry of of y that it sees. If $Ay = 0$, it should return "Zero!" with probability at least $2/3$. If $Ay \neq 0$, it should return "Not Zero!" with probability at least $2/3$.

   (Hint 1: Use part (a). Hint 2: You may use the fact that $\mathbb{F}_{q^s}$ contains $\mathbb{F}_q$ as a subfield for any $s = 1, 2, 3, \ldots$).

   (c) Let $\mathcal{C} = RS_q((1, \gamma, \gamma^2, \cdots, \gamma^{q-2}), q-1, k)$ be a Reed-Solomon code of length $q-1$ over $\mathbb{F}_q$, where γ is a primitive element of $\mathbb{F}_q$. Give a streaming algorithm that takes in an input $c \in \mathbb{F}_q^{q-1}$ in a streaming fashion, and outputs its best guess about whether or not $c \in \mathcal{C}$. (That is, give a streaming algorithm for *error detection*). As in the previous part, your algorithm is allowed to do field arithmetic over arbitrary fields of size $\text{poly}(q)$ is time $O(1)$.

   Your algorithm should use space at most $O(\log(q))$ bits, and spend $O(\text{polylog}(q))$ time per entry of of y that it sees. If $c \in \mathcal{C}$, it should return "True!" with probability at least $2/3$. If $c \notin \mathcal{C}$, it should return "False!" with probability at least $2/3$.

   (Hint: Use part (b)).

4. (**Code it up!**, Class 5) Implement the Berlekamp-Welch algorithm over $\mathbb{F}_{113}$ (or you can implement the Berlekamp-Massey algorithm if you want, or read about any other RS decoding algorithm and implement that). I like Sage (`sagemath.org`) for finite field stuff, but feel free to use whatever language you like; you can even "implement" it by hand if you are very patient. Use your implementation to solve the following decoding problems:

   Consider the Reed-Solomon code $RS_{113}(\{1, \ldots, 16\}, 16, 8)$. The following are codewords with errors in fewer than (half the minimum distance) locations. Find the original codewords.

   (a) `[52, 84, 35, 108, 70, 78, 43, 109, 66, 20, 100, 103, 11, 41, 14, 70]`

   (b) `[7, 64, 58, 10, 90, 89, 99, 54, 42, 55, 82, 24, 35, 95, 38, 25]`

   (c) `[81, 81, 93, 60, 27, 12, 37, 72, 68, 58, 67, 4, 76, 105, 49, 35]`

   If you used something other than the Berlekamp-Welch or Berlekamp-Massey algorithms, explain what you did. If you wrote code, please attach it to your homework. If you did it by hand, show your work. If you did it in your head, I don't believe you.

Note: You are **not** allowed to use existing software packages that implement an RS decoder. However, you are allowed to Google around for Sage technical support, or any other programming help.

Second note: For fun, implement both Berlekamp-Massey and Berlekamp-Welch. Which is faster?

5. (**Decoding RS codes is NP hard**, Class 4) Recall that the problem of *Maximum Likelihood Decoding* for the Reed-Solomon code $\mathcal{C} = RS_q(S, n, k)$ is the following:

   - **Input:** A received word $w \in \mathbb{F}_q^n$ and S, n, k.
   - **Output:** The codeword $c \in \mathcal{C}$ that minimizes $\Delta(w, c)$.

   The "decision version" of this problem (which will call RS-MLD) is the following:

   - **Input:** A received word $w \in \mathbb{F}_q^n$; S, n, k; and $r \in \{1, \ldots, n\}$.
   - **Output:** TRUE if and only if there exists some $c \in \mathcal{C}$ so that $\Delta(w, c) \leq r$.

   We've seen that RS-MLD is easy (meaning, we have polynomial-time algorithms for it) if $r \leq \lfloor (d-1)/2 \rfloor$. In this problem, we'll show that if r is allowed to be very large then RS-MLD is NP-hard.

   (Note: It is possible to do this problem if you have never seen the definition of NP-hardness).

   (a) Let $q = 2^m$ and let $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$. (Notice that because q is a power of 2, $+$ and $-$ are the same in $\mathbb{F}_q$. So I can't make any sign errors in writing the problem and you can't make any when solving it!) Fix $\beta \in \mathbb{F}_q$. Let $y \in \mathbb{F}_q^n$ be the vector so that

   $$y_i = \alpha_i^{k+1} + \beta \cdot \alpha_i^k.$$

   Show that there exists a codeword $c \in RS_q(S, n, k)$ so that $\Delta(c, y) \leq n - k - 1$ if and only if there exists $T \subseteq \{1, \ldots, n\}$ with $|T| = k + 1$ so that $\sum_{i \in T} \alpha_i = \beta$.

   Hint 1: Let $g(X) = X^{k+1} + \beta X^k$. Let $f(X)$ be the polynomial corresponding to the RS codeword c. What can you say about $g(X) - f(X)$? (How many roots does it have? Can you factor it?)

   Hint 2: Elaborating on Hint 1, try to write the coefficient of X^k in $g(X) - f(X)$ in two ways.

   (b) Consider the following problem:

   - **Input:** Parameters k, n so that $1 \leq k < n$, and a set $S \subseteq \mathbb{F}_q$ of size n, and $\beta \in \mathbb{F}_q$.
   - **Output:** TRUE if and only if there exists some $T \subseteq \{1, \ldots, n\}$ of size $|T| = k + 1$ so that $\sum_{i \in T} \alpha_i = \beta$.

   This problem is NP-hard. (You don't have to prove this). **If you have seen the definition of NP-hardness before:** explain why part (a) implies that RS-MLD is NP-hard. **If you have not seen this before:** explain intuitively why this problem being "hard" should imply that RS-MLD is "hard."

6. (**View as cyclic codes**, Class 4) *[This question is long but most of it is exposition; don't get scared.]* In this exercise we'll explore another view of Reed-Solomon codes than we took in class. We say that a code is *cyclic* if we can cyclically shift the entries of a codeword without leaving the code. For example, the code

   $$\mathcal{C} = \{(1, 2, 3), (3, 1, 2), (2, 3, 1)\}$$

   is cyclic. For the rest of this exercise, let $n = q-1$; we'll be considering the code $RS_q(\{\gamma, \gamma^2, \ldots, \gamma^n\}, k)$ for some $k \leq n$.

   (a) Show that $RS_q(\{\gamma, \gamma^2, \ldots, \gamma^n\}, n, k)$ is cyclic, where γ is a primitive element of $\mathbb{F}_q$.

(b) For any $c = (c_0, \ldots, c_{n-1}) \in \mathbb{F}_q^n$, define the polynomial $c(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$. Explain what the polynomial $X \cdot c(X) \mod X^n - 1$ looks like.

(Note: In this context, mod $X^n - 1$ means that you can replace every instance of X^n with 1. For example, $X^{n+5} = X^5 \mod X^n - 1$.) Let $\mathbb{F}_q[X]/(X^n - 1)$ denote the set of polynomials under this equivalence.)

Conclude that for a linear cyclic code $\mathcal{C}$, the set $\{c(X) : c \in \mathcal{C}\} \subseteq \mathbb{F}_q[X]/(X^n - 1)$ is closed under multiplication by X. Further conclude that this set is closed under multiplication by all polynomials $f \in \mathbb{F}_q[X]/(X^n - 1)$.

Terminology: $\mathbb{F}_q[X]/(X^n - 1)$ is a *quotient ring.* In a ring R, a set $I \subset R$ which is closed under addition and also is closed under multiplication by all elements in R (that is, so that $\{rx : \forall r \in R, x \in I\} \subseteq I$) is called an *ideal.*

(c) It turns out (you do not have to prove this) that any set of polynomials in $\mathbb{F}_q[X]/(X^n - 1)$ with the property that you further concluded in (b) has the form

$$\{c(X) : c \in \mathcal{C}\} = \{f(X) \cdot g(X) : f(X) \in \mathbb{F}_q[X]/(X^n - 1)\}$$

for some polynomial $g(X)$. The (unique) polynomial g of minimal degree which is monic (that is, the leading coefficient is 1) is called the *generator polynomial* for $\mathcal{C}$. If the code has dimension k, then the generator polynomial is a monic polynomial with degree $n - k$.

Terminology: $\mathbb{F}_q[X]/(X^n - 1)$ is a *principal ideal ring,* which means that any ideal is generated by a single element: in the same notation as above, there is some $g \in I$ so that $I = \{gx : x \in R\}$.

There is no question for part (c). Just understand that there is such a g. Write "I understand" as the solution. (You don't need to understand the terminology, that's just in case you were interested).

(d) Suppose $\mathcal{C} = RS_q(\{\gamma, \gamma^2, \ldots, \gamma^{q-1}\}, q-1, k)$. Show that the generator polynomial of $\mathcal{C}$ is

$$g(X) = \prod_{i=1}^{n-k} (X - \gamma^i)$$

where γ is a primitive element of $\mathbb{F}_q$.

(Hint: Use the dual view of RS codes that we had in class).
(Another hint: You may use the algebra fact that if a polynomial f has $f(\alpha) = 0$, then $(X - \alpha)$ divides f: that is, we may write $f(X) = (X - \alpha)h(X)$ for some h.)

(e) Finally, let's consider another view of systematic encoding for a linear cyclic code $\mathcal{C}$ of dimension k and length n with generator polynomial $g(X)$. Let $m = (m_0, \ldots, m_{k-1})$ be a message to encode. Suppose we wish to find a systematic encoding

$$m \mapsto (p_0, \ldots, p_{n-k-1}, m_0, m_1, \ldots, m_{k-1}).$$

(Notice the message symbols are at the end rather than the beginning; this will be a bit more convenient). Argue that the parity-check symbols $p_0, \ldots, p_{n-k-1}$ are given by the coefficients of a polynomial $p(X)$ so that

$$p(X) = g(X)a(X) - X^{n-k}m(X) \mod X^n - 1$$

for some polynomial $a(X)$.

(f) Show that we can compute the parity bits of a code $\mathcal{C}$ as in part (e) by computing

$$p(X) = -\text{remainder}\left(\frac{X^{n-k}m(X)}{g(X)}\right),$$

where remainder just means as in normal polynomial division.

(Hint/reminder: The guarantee of polynomial division is the following. For any pair of polynomials $a(X), b(X)$, there are unique polynomials $q(X)$ and $r(X)$ so that the following hold: either $r(X) = 0$ or else $\deg(r) < \deg(b)$, and $a(X) = b(X) \cdot q(X) + r(X)$.)

Section 3

1. Lagrange interpolation gives a formula to find $f(\beta)$ given evaluations of $f(\alpha_i)$ for $i = 1, \ldots, k$. How long (using big-oh notation) does it take to evaluate this formula, and actually find $f(\beta)$ given $\{f(\alpha_i) : i = 1, \ldots, k\}$? Call that time $T(n, k)$. What if you want to find $f(\beta_j)$ for $j = 1, \ldots, \ell$; that is, if you'd like to find multiple points $f(\beta)$ at once? Obviously you can do this in time $\ell \cdot T(n, k)$ by repeating the procedure above ℓ times, but can you do better when ℓ gets large? How large does ℓ have to be before meaningful improvements kick in?

2. What is the smallest t-wise independent source that exists over $\mathbb{F}_2$? Did you find it in Section 2, Problem 1, or can you do better?

3. We've seen in class that there's no point in considering polynomials over $\mathbb{F}_q$ of degree larger than $q-1$. For example, $f(X) = X^q$ has the same behavior (in terms of evaluations over $\mathbb{F}_q$) as $f(X) = X$. However, suppose we also include some derivatives: consider the code

$$\{\langle (f(\alpha_1), f'(\alpha_1)), (f(\alpha_2), f'(\alpha_2)), \ldots, (f(\alpha_n), f'(\alpha_n)) \rangle : \deg(f) < k\},$$

   which is a code of length n over an alphabet of $\mathbb{F}_q^2$. Here, f' is just the formal derivative of f, gotten by linearly extending the operation $d/dX X^i = iX^{i-1}$. What can you say about this code? Now is it sensible for polynomials to have degrees larger than $q-1$? What's the rate and distance of this code? (This is called a *derivative code* or a *univariate multiplicity code.* It may show up again in the context of list and local decoding).

4. In class we defined BCH codes. Consider the family of *dual BCH codes,* which as you might expect are defined to be the dual codes of BCH codes.

   (a) Let $q = 2^t$. The **field trace** from $\mathbb{F}_q$ to $\mathbb{F}_2$ is defined by $\mathrm{tr}(X) = X + X^2 + X^{2^2} + \cdots + X^{2^{t-1}}$. It turns out (fun exercise, prove this) that $\mathrm{tr} : \mathbb{F}_q \to \mathbb{F}_2$ (aka, the image is $\mathbb{F}_2$), and moreover $\mathrm{tr}(X)$ is $\mathbb{F}_2$-linear, in the sense that $\mathrm{tr}(x + y) = \mathrm{tr}(x) + \mathrm{tr}(y)$. Show that the dual BCH codes can also be defined as the family of codes

$$\mathrm{DualBCH}_{q,d} = \left\{(\mathrm{tr}(f(1)), \mathrm{tr}(f(\gamma)), \ldots, \mathrm{tr}(f(\gamma^{q-2}))) \ : \ f : \mathbb{F}_q \to \mathbb{F}_q, \deg(f) \leq k\right\}.$$

   (That is, show that this code is dual to the BCH codes we saw in class, for an appropriate choice of k). This shows that the duals of BCH codes are just the traces of RS codes!

   (b) Can you come up with an efficient decoding algorithm for these codes?

5. In section 2 we saw that it is NP-hard to do Maximum Likelihood Decoding (MLD) on Reed-Solomon codes in general—that is, given a worst-case $w \in \mathbb{F}^n$, it's probably hard to find the nearest RS codeword. However, as we have seen, if we are promised that the number of errors are less than half the distance, we can efficiently find the (unique) nearest codeword. We will see later in the course that if we are promised a number of errors is slightly larger (as large as the "Johnson radius") we can also implement MLD efficiently. What's the biggest radius you can find where you can do MLD efficiently, before the problem becomes NP hard? How about before it is hard under stronger complexity assumptions (eg, "factoring is hard?")