Chinese Remainder Codes

Milan Mossé

March 2019

Introduction

These notes introduce and discuss Chinese Remainder Codes. The first section introduces the Chinese Remainder Theorem for $\mathbf{Z}/n\mathbf{Z}$ and the basics of Chinese Remainder Codes. The second section discusses algorithms for unique and list-decoding Chinese Remainder Codes. The third introduces a generalized form of the Chinese Remainder Theorem and of Chinese Remainder Codes.

This document aims to give the reader with little or no relevant background the tools to think and read more about Chinese Remainder Codes, but my hope is that it's also helpful to people who have some background. The end of the first and final chapters gives a couple of exercises for the reader, but this document is primarily concerned with collecting together, in a readable way, information from different sources on this topic. For a very clear and concise introduction to Chinese Remainder Codes, with a focus on a few recent papers, see [2].

1 Chinese Remainder Codes

1.1 Chinese Remainder Theorem

If you run around a track, you'll find that the numbers circle around: 100m, 200m, 300m, 400m, 100m, ... And the same is true for time: in the United States, our hours go $1, 2, 3, ..., 12, 1...$, and in most other places, they instead go up to 24. People are familiar with arithmetic that is circular or "ring-shaped" in this way: if I ask you what time it will be 8 hours after 10pm, (if you live in the US) you will add $8+10 = 18$ and subtract 12 to get 6am. It's useful to have symbols to refer to this kind of arithmetic, so using $\mathbf{Z}$ to denote the integers, we'll use $\mathbf{Z}/12\mathbf{Z}$ to denote *the integers modulo (mod) 12.* This gives us a first example:

Example 1.1.1. For two numbers n, m in $\mathbf{Z}/12\mathbf{Z}$, we calculate their sum or their product as usual and then subtract 12 until we get a positive number less than 12. For example, $8 + 10 = 6 \pmod{12}$, and $8 * 11 = 4 \pmod{12}$.

Now, we can analogously define $\mathbf{Z}/n\mathbf{Z}$ for any integer n:

Definition 1.1.2. Fix an integer n. The *integers mod n*, denoted $\mathbf{Z}/n\mathbf{Z}$, is the set[1] of numbers (called *residues*, *representatives*, or *equivalence classes*) $0, 1, 2, ..., n-1$, with addition and multiplication defined by taking the remainder upon division by n.

The focus of this section will be a theorem that will help us to understand the additive and multiplicative structure of $\mathbf{Z}/n\mathbf{Z}$. We'll consider an example that will serve to motivate the theorem, and then we'll prove it.

Example 1.1.3. Consider the residues of $\mathbf{Z}/12\mathbf{Z}$:

$$0, 1, 2, 3, ..., 11.$$

Note that each of these correspond to a unique pair of residues in $\mathbf{Z}/3\mathbf{Z}$ and $\mathbf{Z}/4\mathbf{Z}$. In a table:

$\mathbf{Z}/12\mathbf{Z}$	$\mathbf{Z}/3\mathbf{Z}$	$\mathbf{Z}/4\mathbf{Z}$
0	0	0
1	1	1
2	2	2
3	0	3
4	1	0
5	2	1
6	0	2
7	1	3
8	2	0
9	0	1
10	1	2
11	2	3

The key observation here is that each number on the left-hand-side (LHS) gives a unique pair of residues on the RHS.

Let $n = 12$, $n_1 = 3$, and $n_2 = 2^2 = 4$. (Note that $n = n_1n_2$.) The possibility of putting congruences (mod n) into correspondence with congruences (mod n_1) and (mod n_2), as in the above example, depends crucially on the fact that n_1, n_2 share no common factors. We give this property a special name:

Definition 1.1.4. Integers n, m are *coprime* if they share no prime factors.

Now, we can generalize from the preceding example (for this formulation of the proof we draw on [8]):

Proposition 1.1.5. *Suppose that n_1, n_2 are coprime positive integers and let $n = n_1n_2$. Then given residue classes a_1 (mod n_1) and a_2 (mod n_2), there is a unique residue class a (mod n) such that*

$$a \equiv a_1 \ (mod\ n_1), \ \ and\ a \equiv a_2 \ (mod\ n_2).$$

[1]One can instead define $\mathbf{Z}/n\mathbf{Z}$ using the notion of an *equivalence class,* but my hope is that this definition is friendlier to someone who is new to this.

In other words, there exists a solution to the pair of congruences a_1 (mod n_1) and a_2 (mod n_2), and all the solutions to these congruences form a residue (mod $n_1 n_2 = n$).

Proof. First, we show that if x, y satisfy both congruences, they are equal (mod n), and that any residue equal to these (mod n) satisfies both congruences. Indeed, we have

$$x \equiv y \,(\text{mod } n_i),$$

so that n_i divides $x - y$. Since the n_i are coprime, n divides $x - y$. Thus the above equivalence holds mod n. And if x is a solution, then $x + kn_1n_2$ is also a solution to both congruences.

Now, we show that there exists a solution to these two congruences. Since n_1, n_2 are coprime, there exist integers k, ℓ such that

$$n_1 k + n_2 \ell = 1.$$

Then

$$x = a_2 n_1 k + a_1 n_2 \ell$$

is a solution. Indeed,

$$x \equiv a_1 n_2 \ell \equiv a_1(1 - n_1 k) \equiv a_1 \,(\text{mod } n).$$

Symmetrically, $x \equiv a_2 \pmod{n_2}$. □

The sets

$$\mathbf{Z}/n\mathbf{Z}, \text{ and } \mathbf{Z}/n_1\mathbf{Z} \times \mathbf{Z}/n_2\mathbf{Z}$$

are equal in size; the above theorem gives a particular bijection ψ between them, which maps a residue on the LHS to two residues in the components on the RHS. The Chinese Remainder Theorem simply states that this function ψ preserves structure:

Theorem 1.1.6 (The Chinese Remainder Theorem for $\mathbf{Z}/n\mathbf{Z}$). *Let $n = n_1 n_2$, with n_1, n_2 coprime. Let a, b be residues (mod n). The function ψ which maps a (mod n) to the pair (a_1, a_2) of its representatives (mod n_1) and (mod n_2) preserves structure:*

$$\begin{aligned} \psi(ab) &= \psi(a)\psi(b), \text{ and} \\ \psi(a+b) &= \psi(a) + \psi(b), \end{aligned}$$

where we define multiplication and addition on the pairs of congruences componentwise. (That is, $(a_1, a_2) * (b_1, b_2) = (a_1 * b_1, a_2 * b_2)$, where $*$ is multiplication or addition.)

Having shown that ψ is a bijection, we leave the proof of the above theorem as an exercise. There are simpler ways of stating this theorem, which we will record for future reference. To do that, we need two definitions:

Definition 1.1.7. A tuple $(R, +, \times)$ (often just written R) is called a *ring* if the operation $+$ (*addition*) is associative and commutative with inverses and identity; the operation $\times$ (*multiplication*) is associative with identity; and multiplication distributes over addition. If multiplication is commutative, the ring is called a *commutative ring.*

Example 1.1.8. The integers $\mathbf{Z}$, rationals $\mathbf{Q}$, and reals $\mathbf{R}$ are rings.

Definition 1.1.9. A function on rings $\psi : R_1 \to R_2$ is called a *ring homomorphism* if it preserves structure, i.e., if

$$\begin{aligned} \psi(ab) &= \psi(a)\psi(b), \text{ and} \\ \psi(a+b) &= \psi(a) + \psi(b). \end{aligned}$$

If ψ is also a bijection, it is then called a *ring isomorphism.*

Now, we can restate Theorem 1.6:

Corollary 1.1.10. *The bijection ψ is a ring isomorphism, and $\mathbf{Z}/n\mathbf{Z}$ and $\mathbf{Z}/n_1\mathbf{Z} \times \mathbf{Z}/n_2\mathbf{Z}$ are isomorphic. Notationally,*

$$\mathbf{Z}/n\mathbf{Z} \cong \mathbf{Z}/n_1\mathbf{Z} \times \mathbf{Z}/n_2\mathbf{Z}.$$

Note that, in particular, if n can be factored $p_1^{k_1} \cdots p_n^{k_n}$, with the p_i all distinct, repeated application of the above corollary gives the following result:

Corollary 1.1.11. *The following isomorphism holds:*

$$\mathbf{Z}/n\mathbf{Z} \cong \mathbf{Z}/p_1^{k_1}\mathbf{Z} \times \cdots \times \mathbf{Z}/p_n^{k_n}\mathbf{Z}$$

To conclude our discussion of the Chinese Remainder Theorem, in case it's of interest, we mention that most of the above theorem can be shown using Galois theory. (If you're not interested or haven't seen Galois theory before, you can safely skip the rest of this subsection.)

We'll need to introduce a few propositions.

Proposition 1.1.12. *If K_1, K_2 are Galois extensions of a field F that is also their intersection, then*

$$Gal(K_1K_2/F) \cong Gal(K_1/F) \times Gal(K_2/F).$$

(For a proof, see p. 593 of [1].) Let ζ_n be the n^{th} root of unity, and express $n = p_1^{k_1} \cdots p_n^{k_n}$. Using the fact that each of the fields $\mathbf{Q}(\zeta_{p_i^{k_i}})$ are contained inside $\mathbf{Q}(\zeta_n)$, one can use the above proposition to establish the following:

Proposition 1.1.13. *Express $n = p_1^{k_1} \cdots p_n^{k_n}$. Then*

$$Gal(\mathbf{Q}(\zeta_n)/\mathbf{Q}) \cong Gal(\mathbf{Q}(\zeta_{p_1^{k_1}})/\mathbf{Q}) \times \cdots \times Gal(\mathbf{Q}(\zeta_{p_n^{k_n}})/\mathbf{Q})$$

(For a proof, see pp. 598-9 of [1].) We introduce a final proposition:

Proposition 1.1.14. *Let ζ_n be the n^{th} root of unity. Then*

$$(\mathbf{Z}/n\mathbf{Z})^\times \cong Gal(\mathbf{Q}(\zeta_n)/\mathbf{Q}).$$

(For a proof, see pp. 596-7 of [1].) Then, using the last two propositions, we obtain the part of the Chinese Remainder Theorem for $\mathbf{Z}/n\mathbf{Z}$ that concerns multiplicative groups:

Corollary 1.1.15. *The following isomorphism holds:*

$$(\mathbf{Z}/n\mathbf{Z})^* \cong (\mathbf{Z}/p_1^{k_1}\mathbf{Z})^* \cdots \times (\mathbf{Z}/p_n^{k_n})^*.$$

1.2 Chinese Remainder Codes

Now, suppose that you want to send a message of length K to someone, but you suspect that, (for example, because someone could intercept your message), some letters could be erased, changed, or deleted in your message. For example, if $K = 14$, your message could be "CRCs are cool!" and you might worry that it could be turned into something like

"CRCsYare cool!", "CRCs _re cool!", or "CRCs ae cool!".

The first of these is called an *error*, the second an *erasure*, and the third a *deletion*. In this document, we'll be focusing on errors and erasures, and especially on errors. Chinese Remainder Codes provide a way of encoding your message m of length K into a *codeword* of length $N \geq K$, that allows the receiver of the codeword to correct for errors and erasures.

To see how this works, we need to cast your message in terms of modular arithmetic. To do this, we need some basic set-up. Fix a positive integer n. Let $p_1 < \cdots < p_n$ denote positive integers which are relatively prime (for example, the first n primes). Fix $k \leq n$. Let $K = p_1 \cdots p_k$ and $N = p_1 \cdots p_n$. Now, we can show that m can be represented by an equivalence class in $\mathbf{Z}/K\mathbf{Z}$:

Proposition 1.2.1. *Your message m with k letters can be represented by an equivalence class in $\mathbf{Z}/K\mathbf{Z}$.*

Proof. The message has k letters. We impose a restriction on the message: the i^{th} letter must be represented by an integer mod p_i. Then, we see that your message

$$m = (m_1, ..., m_k) \text{ is in } \mathbf{Z}/p_1\mathbf{Z} \times \cdots \mathbf{Z}/p_k\mathbf{Z},$$

so that by Proposition 1.1.5, we can say (abusing notation) that m is in $\mathbf{Z}/K\mathbf{Z}$. We now see that this restriction is not all that harmful: we can use a look-up table to correlate each of the numbers between 1 and K (inclusive) with a distinct message over an arbitrary alphabet. If the alphabet has ℓ letters, then m will have a length over that alphabet of $\log_\ell(K)$. ∎

Chinese Remainder Codes use the above fact to encode a message:

Definition 1.2.2. Fix a message m in $\mathbf{Z}/K\mathbf{Z}$. We define an *encoding map* ENC that sends the message m to a codeword:

$$ENC(m) = (m_1, ..., m_n), \text{ where } m \equiv m_i \pmod{p_i}, \text{ for all } i \text{ in } [n].$$

We call the image $ENC(\mathbf{Z}/K\mathbf{Z}) = CRC_{p_1,...,p_n;k}$ the *Chinese Remainder Code over* $p_1, ..., p_n$*, with message length* k*.* With sufficient context, we simply denote the code *CRC*. We write $\bar{m}$ for $ENC(m)$.

In the remainder of this section, we'll establish some basic properties of this code. The first thing we might want to know is whether ENC and ENC^{-1} are efficiently computable—that is, computable in polynomial time:

Remark 1.2.3. It turns out that ENC and ENC^{-1} are both efficient (and practically) computable. As a first pass, consider the task of computing m mod p_i. This can be done (n times) to encode m. If we naively subtract p_i from m until we get to a number between 1 and p_i, this takes $\Theta(m/p_i)$ steps.[2] Doing this for each of the n values of i gives a total of

$$\sum_{i=1}^{n} \Theta(m/p_i) \sim nO(K/p_n) \sim O(nK/p_n) \sim O(K) \text{ steps,}^{3}$$

assuming n is large. So naive computation of ENC is "efficient" (linear in K), but if K is large this could in practice take a very long time, especially if we want to encode messages several (for example, K) times.

Let us now consider how this could be made faster in practice. Intel processors compute the modulus in a single instruction for values up to 64 bits, but practically, the primes p_i may be larger than that. For example, if you want to send a key for an encryption algorithm, it should be at least 2048 bits in length. There are efficient algorithms for calculating mod larger numbers (see [9]), but the fastest thing, if we fix N and K in advance, is to use a lookup table. Then ENC and ENC^{-1} can each be computed in constant time.[4]

Now, our second task will be to see how one can use this code to correct for errors and erasures. Consider the encoding of m

$$\bar{m} = (m_1, ..., m_n).$$

Suppose $\bar{r} = (r_1, ..., r_n)$ is the corruption of $\bar{m}$. We would like to use the fact that $\bar{m}$ is in the CRC to correct $\bar{r}$. We would like a basic metric of how far apart $\bar{r}$ and $\bar{m}$ are (we will improve this metric shortly):

Definition 1.2.4. Fix vectors $\bar{m}, \bar{r}$. The *distance* between them is the number of indices at which they are unequal. The *distance d of a code* is the minimum distance between all of the codewords in the code.

[2]This is a fancy way of saying that this takes more than $c(m/p_i)$ and less than $C(m/p_i)$ steps to compute, for some constants $c \leq C$.

[3]This is a fancy way of saying that there is a constant C such that this takes at most CnK/p_n steps,

[4]Thanks to to Noah Shutty for much help in thinking through the efficiency of these maps.

If all of our codewords are at least d apart, then if $\bar{r}$ has at most $\lfloor (d-1)/2 \rfloor$ errors or $d-1$ erasures, we can correct it by finding the (unique) closest CRC codeword. Conversely, if our code can correct at most $\lfloor (d-1)/2 \rfloor$ errors, that it has distance d. To consider how good the distance of the CRC is, we consider a bound[5] on any code's distance:

Proposition 1.2.5 (The Singleton Bound)**.** *Fix a code $\mathcal{C}$ with message length k, codeword length n, and minimum distance d between codewords. Then $k \leq n-d+1$ (equivalently: $d \leq n-k+1$).*

Proof. Consider the set $\tilde{\mathcal{C}}$ of codewords in $\mathcal{C}$, with the last $d-1$ coordinates deleted. Then $|\tilde{\mathcal{C}}| = |\mathcal{C}|$; since the minimum distance is d, none of the truncated codewords are the same. Let q be the number of letters in the alphabet of $\mathcal{C}$. Then $|\mathcal{C}| = q^k$, since there is one codeword for every message we can send.

Since $|\tilde{\mathcal{C}}| \leq q^{n-d+1}$, these two equalities give

$$q^k \leq q^{n-d+1} \implies k \leq n-d+1,$$

as required. □

It's a nontrivial feat for a code to meet the Singleton Bound, so we give the codes that do a special name:

Definition 1.2.6. A code $\mathcal{C}$ that meets the Singleton Bound ($k = n-d+1$) is called *Maximum Distance Separable,* or MDS.

Chinese Remainder Codes are MDS! Let us see why:[6]

Theorem 1.2.7. *The CRC over $p_1, ..., p_n$ with message length k is MDS.*

Proof. We must show that $d \geq n-k+1$, and that is just to say that any distinct messages $\bar{m}, \bar{m'}$ must have less than k indices at which they agree. Let $A(\bar{m}, \bar{m'})$ (the *agreement*) be the set of indices at which they differ. We must show that $|A(\bar{m}, \bar{m'})| < k$. Since, for all i in A,

$$m \equiv m' \pmod{p_i},$$

we have that

$$P_A := \prod_{i \in A} p_i \text{ divides } m - m'.$$

Thus, $P_A \leq |m - m'| < K$, since m and m' are distinct and at most K. So $|A| < k$. □

In a sense, this is satisfying (we met the bound!), but it isn't entirely clear what it would mean to say "our code can correct such-and-such-many errors."

[5]due to [7], but for this formulation of the proof, we follow [10].

[6]For the proof, this way of framing the problem, and the brief discussion that follows, we follow [10].

Because the letters of the codeword m_i are all of different lengths, correcting m_1 is an easier task than correcting m_n.

To see this, take a toy example: suppose we have a string xy of length two, where x is in the alphabet $\{0, 1\}$, and y is in the alphabet $\{1, ..., 2^{100}\}$. It's a bit deceptive to say of any code that it can correct "one error" of such a string, because there's so much more information being stored in y than in x! We need a metric to measure *the amount of information* the CRC can recover, and we've seen that the amount of errors don't measure this precisely. One solution is to pick the coprime numbers p_i so that they're very close together; then, each letter would carry a similar amount of information. But this is not always easy, especially if we're picking the numbers p_i to be prime, since we would we need to know more about how "close together" primes can be, in order for us to be able to pick some that are close together. It is better then, to change the way that we're considering errors: we can weigh each error by the alphabet it came from. Formally, we make explicit and develop the definitions tacit in the above proof:

Definition 1.2.8. The *agreement* $A(\bar{m}, \bar{r})$ is the set of indices at which $\bar{m}, \bar{r}$ agree. The *disagreement* $D(\bar{m}, \bar{r})$ is the set of indices at which they disagree.

Definition 1.2.9. For any subset S of $[n]$, define $P_S = \prod_{i \in S} p_i$.

Definition 1.2.10. The *amplitude of agreement* is P_A. The *amplitude of disagreement* is P_D.

Now, we can state our desiderata for the CRC. We would like for it to be able to correct errors and erasures, where the corrupted codeword $\bar{r}$ has *high amplitude of disagreement* from our message $\bar{m}$. Then, we can "translate" the Singleton Bound, using this new metric. First, we translate distance:[7]

Definition 1.2.11. Say that a CRC has *amplitude distance* D if this is is the largest value for which, given any agreement between codewords $A = A(\bar{m}, \bar{m'})$, we have

$$D \leq N/P_A.$$

As D increases, amplitude of agreement (P_A) gets smaller between codewords. Now, we can state the Singleton Bound for amplitudes:

Theorem 1.2.12 (Singleton Bound for Amplitudes)**.** *Suppose we are given a code, possibly with different alphabets for different letters. For codewords, let N be the product of the sizes of the alphabets for each letter. For messages, let K be the product of the sizes of the alphabets for letters. For a code with amplitude distance D, we have $D \leq N/K$.*

Note that if all n letters in the codeword and all k in the message come from an alphabet of size q, and the distance is d, this says that $q^d \leq q^{n-k}$, implying the Singleton Bound from above. And the CRC meets the Singleton Bound for

[7]For this formulation, we draw again on [10].

Amplitudes, as well. Two exercises that may be useful to the reader are proving the above theorem and that the CRC meets it. (The resources to do this are contained in this section.)

In the next section, we consider efficient algorithms for decoding CRCs. It turns out that even though they're MDS and efficiently encodable and decodable, CRCs aren't used much in the real world. When it comes to encoding, for example, CDs, people tend to use different (also MDS, also efficiently encodable and decodable) codes called *Reed Solomon (RS) Codes.* This raises a simple quesiton: *why not use CRCs, if they have many of the same benefits as Reed Solomon Codes?* The most concise answer is that RS codes can have alphabets of length 2^i for arbitrary i, and CRCs, as we have seen, cannot: because data is stored in binary, it's useful to have an alphabet with length a power of 2.

2 Decoding Algorithms

We saw earlier that any code with distance d can correct up to $\lfloor (d-1)/2 \rfloor$ errors: find the unique closest codeword, and return it. However, looking through *all* codewords can take a lot of time, and one would prefer to have more efficient decoding algorithms. In this section, we discuss in detail[8] a unique decoding algorithm, due to [4] that corrects any error amplitude less than E as long as $E^2 \leq N/K$. This implies a bound on the number of errors e the algorithm can correct:

$$e \leq (n-k) \cdot \frac{\log p_1}{\log p_1 + \log p_n}.$$

Note that from our discussion of the Singleton Bound for Amplitudes, we know that this bound on E is not optimal. We refer the reader to reading on improvements on this bound and more general results from the literature at the end of this section.

Now, we describe the algorithm. The algorithm takes as input the corrupted codeword $\bar{r} = (r_1, ..., r_n)$, and outputs the unique message m such that $P_{A(\bar{r},\bar{m})} \geq N/E$, where $E^2 < N/(K-1)$. It has three steps:

1. Find r, using $\bar{r}$.
2. Find integers y and z such that
$$1 \leq y \leq E,\ 0 \leq z \leq N/E, \text{ and } y \cdot r \equiv z \pmod{N}.$$
3. Output z/y if it's an integer.

First, let check that this algorithm is efficient. The first step is efficient, because to do it, we apply ENC^{-1} to $\bar{r}$, and that is efficient. The second step is an instance of a problem called *Integer Linear Programming* in a fixed number of

[8] Our discussion follows [2], but we made some adjustments for clarity, discussed above in more detail the efficiency of encoding and decoding, and we discuss in more detail the error bound.

variables, and a discussion of why this is efficient would take us too far afield; we refer the reader to [5] for discussion.

Now, we show that this algorithm works. We must show that there exist y, z satisfying the above relations, and that $z/y = m$ for any such y, z. Let us prove each of these in turn. Recall that we assumed that the amplitude of error $P_{D(\bar{r},\bar{m})}$ is less than E. We give this fact a name (call it $(*)$) so that we can refer to it later.

Theorem 2.0.1. *There exist integers y and z such that*

$$1 \leq y \leq E,\ 0 \leq z \leq N/E, \ \ \textit{and}\ y \cdot r \equiv z\ (\textit{mod}\ N).$$

Proof. Pick any such y, and we show that there exists such a z. Let y be the amplitude of disagreement between $\bar{r}$ and $\bar{m}$. By $(*)$, we have $y \leq E$. Note also that

$$y \cdot r \equiv y \cdot m \pmod{p_i}, \text{ for all } i,$$

since if i is a point of disagreement, this is 0, and otherwise it is a point of agreement, and equality again holds. Using 1.1.5, it follows that $y \cdot m \equiv z$ (mod N), for some z. Since $y \leq E$, $m \leq K - 1$, and we have $E < \sqrt{N/(K-1)}$, it follows that $z < N/E$, as required. □

Now, we prove that $z/y = m$ for any such y, z.

Theorem 2.0.2. *Consider any y, z that satisfy*

$$1 \leq y \leq E,\ 0 \leq z \leq N/E, \ \ \textit{and}\ y \cdot r \equiv z\ (\textit{mod}\ N).$$

Then if z/y is an integer, it is equal to z.

Proof. For any i in $A = A(\bar{r}, \bar{m})$, we have that

$$z \equiv y \cdot r \equiv y \cdot m \pmod{p_i},$$

so by 1.1.5, we have $y \cdot m \equiv z$ (mod P_A). It will suffice to show that $y \cdot m$ and z are both less than P_A. By $(*)$, we have $P_A > N/E$. So, we can just show that these are less than N/E, and this was shown at the end of the previous proof. □

So, our algorithm works and is efficient! To get the bound on the number e of errors we can tolerate, just note that $E^2 \leq N/k$ gives

$$\prod_{i=n-e+1}^{n} p_i^2 \leq \prod_{i=n-k+1}^{n} p_i.$$

This holds if $p_n^e < p_1^{n-(e+k)}$, so rearranging gives

$$e \leq (n-k) \cdot \frac{\log p_1}{\log p_1 + \log p_n},$$

which is the bound stated at the start of this section.

This concludes our discussion of the unique decoding algorithm for errors. This approach can be improved upon (see [2]) and generalized to erasure decoding (see [6]) and other error models (see [3]). In the next section, we introduce generalizations of Chinese Remainder Codes and of the Chinese Remainder Theorem.

3 Generalized Chinese Remainder Codes

This section introduces a generalized version of the Chinese Remainder Theorem, and of Chinese Remainder Codes. Their basic properties are discussed further in [11]; this is just to give the reader a sense for how they are supposed to work.

3.1 Chinese Remainder Theorem Revisited

Recall our earlier formulation of the Chinese Remainder Theorem for $\mathbf{Z}/n\mathbf{Z}$:

Corollary 3.1.1. *Express $n = p_1^{k_1} \cdots p_n^{k_n}$. Then*

$$\mathbf{Z}/n\mathbf{Z} \cong \mathbf{Z}/p_1^{k_1}\mathbf{Z} \times \cdots \times \mathbf{Z}/p_n^{k_n}\mathbf{Z}.$$

In this section, we introduce the Chinese Remainder Theorem in full generality, which implies the above as a corollary. To do this, we need to recall an earlier definition and introduce a few others:

Definition 3.1.2. A tuple $(R, +, \times)$ (often just written R) is called a *ring* if the operation $+$ (*addition*) is associative and commutative with inverses and identity; the operation $\times$ (*multiplication*) is associative with identity; and multiplication distributes over addition. If multiplication is commutative, the ring is called a *commutative ring.*

Definition 3.1.3. An *ideal* is a subset I of a ring R that is closed under addition between elements of I and multiplication of elements of I by elements of R.

Example 3.1.4. Fix an integer n. The set $n\mathbf{Z}$ of all multiples of n is an ideal in the ring $\mathbf{Z}$. When $n = 0$, the ideal is trivially small, since it contains only 0, and when $n = 1$, the ideal is trivially large, since it contains all of $\mathbf{Z}$. (These last two observations hold for any ring R.)

Definition 3.1.5 (Operations on Ideals)**.** The *product* $I_1 \cdots I_n$ of ideals $I_1, ..., I_n$ is the ideal of all finite sums of elements of the form $x_1 \cdots x_n$ with x_i in I_i. The *sum* $I_1 + \cdots + I_n$ is the ideal of all sums of the form $x_1 + \cdots + x_n$ with x_i in I_i.

Definition 3.1.6. Two ideals I_1, I_2 of a ring R are *comaximal* or (and this means the same thing) *relatively prime* when $I_1 + I_2 = R$.

Now, we can state and prove the theorem, which will allow us to introduce a generalized version of Chinese Remainder Codes:[9]

Theorem 3.1.7 (The Chinese Remainder Theorem)**.** *Let $I_1, ..., I_n$ be ideals in R. The map*

$$\psi : R \to R/I_1 \times \cdots \times R/I_n$$

defined by

$$r \mapsto (r + I_1, ..., r + I_n)$$

is a ring homomorphism with kernel

$$\ker \psi = I_1 \cap \cdots \cap I_n.$$

If for each $i \neq j$ the ideals I_i, I_j are comaximal, then ψ is surjective and $I_1 \cap \cdots \cap I_n = I_1 \cdots I_n$, so that

$$R/(I_1 \cdots I_n) = R/(I_1 \cap \cdots \cap I_n) \cong R/I_1 \times \cdots R/I_n.$$

(For a proof and discussion, see pp. 265-7 of [1].) The Chinese Remainder Theorem for integers is obtained as a special case by taking $R = \mathbf{Z}/n\mathbf{Z}$, and $I_i = p_i^{k_i}$, where $n = p_1^{k_1} \cdots p_n^{k_n}$.

3.2 Generalized Chinese Remainder Codes

We can now use the Generalized Chinese Remainder Theorem to introduce *Generalized Chinese Remainder Codes*, defined for *arbitrary* commutative rings R (not just $\mathbf{Z}/N\mathbf{Z}$) and for *arbitrary comaximal ideals* $I_1, ..., I_n$ (not just $\mathbf{Z}/p_i\mathbf{Z}$ with $N = p_1 \cdots p_n$ and n_1, n_2 coprime). Let us consider an example of choices for $R, I_1, .., I_n$:

Example 3.2.1. We can take $R = F[x]$, the polynomial ring with coefficients in a field F (such as $\mathbf{Q}$ or $\mathbf{R}$), and we can take I_i to be the ideal of multiples of some irreducible polynomial $f_i(x)$ of degree i.

Now, we can finally introduce Generalized Chinese Remainder Codes:

Definition 3.2.2. Let R be a commutative ring with identity. Let $I_1, ..., I_s$ be relatively prime ideals of R. Let I^* be the intersection of all of these ideals, let $p_i = |R/I_i|$, and let

$$J_i = I_1 \cap \cdots \cap I_{i-1} \cap I_{i+1} \cap \cdots \cap I_s.$$

Then, for a message m in R/I^*, we use the encoding:

$$ENC(m) = (m_1, ..., m_s), \text{ where } m \equiv m_i \pmod{J_i}.$$

We call the image $ENC(R/I^*)$ the *Generalized Chinese Remainder Code GCRC* over $R, I_1, ..., I_s$.

We direct the reader to [11] for further discussion. There, the reader will find discussion of the properties of GCRCs and proofs (some of which use 3.1.7!) that they have them.

[9]This proof closely follows pp. 265-6 of [1].

4 Conclusion

In the first section, we introduced the Chinese Remainder Theorem and Chinese Remainder Codes, and discussed a few of their basic properties. Then, we discussed an algorithm for efficient decoding of errors. Finally, we saw generalizations of the Chinese Remainder Theorem and of Chinese Remainder Codes. I hope you've enjoyed reading this! If you have any comments, feedback, or ideas for how this could be improved, please send me an email! (You can do that by clicking on my name, at the top of the document.)

References

[1] Dummit, David and Richard Foote (2000). *Abstract Algebra.* Wiley.

[2] Ragesh Jaiswal (draft). "Chinese Remainder Codes: Using Lattices to Decode Error Correcting Codes Based on Chinese Remaindering Theorem."

[3] Guruswami, Sahai, Sudan (2000). "'Soft Decision' Decoding of Chinese Remainder Codes."

[4] Goldreich, Ron, Sudan (2000). "Chinese Remaindering with Errors."

[5] Lenstra, H.W. (1983). "Integer Programming with a Fixed Number of Variables."

[6] Li, Wenhui and Vladimir Sidorenko (2012). "On the Error-Erasure-Decoder of the Chinese Remainder Codes."

[7] Singleton, R.C. (1964). "Maximum Distance q-nary Codes." *IEEE Trans. Inf. Theory*, 10 (2): 116-118.

[8] Soundararajan, Kannan (2017). *Modern Mathematics: Discrete Methods.* Taught at Stanford University.

[9] Will, Mark and Ryan Ko (2014). "Computing Mod Without Mod." Cyber Security Lab, University of Waikato.

[10] Wootters, Mary (2019). *Algebraic Error Correcting Codes.* Course Notes. Taught at Stanford University.

[11] Zhang, Liu (2006). "Chinese Remainder Codes."