General Errors, Uncountability, Self Reference, Counting 5

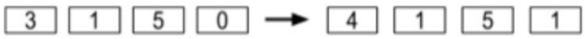
Computer Science Mentors 70

February 26-March 2, 2018

1 General Errors (Berlekamp and Welch)

1.1 Introduction

Now instead of losing packets, we know that k packets are corrupted. Furthermore, we do not know which k packets are changed. Instead of sending k additional packets, we will send an additional $2k$.



Solomon-Reed Codes

1. Identical to erasure errors: Alice creates $n-1$ degree polynomial $P(x)$.

$$P(x) = p_0 + p_1 x + \ldots + p_{n-1}x^{n-1}$$

2. Alice sends $P(1), \ldots, P(n+2k)$

3. Bob receives $R(1), \ldots, R(n+2k)$

For how many points does $R(x) = P(x)$?

Solution: $n + k$

True or false: $P(x)$ is the unique degree $n-1$ polynomial that goes through at least $n+k$ of the received points.

Solution: True

Write the matrix view of encoding the points $P(1), \ldots, P(n+2k)$

Solution:

$$\begin{bmatrix} P(1) \\ P(2) \\ P(3) \\ \vdots \\ P(n+2k) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1^2 & \cdot & 1 \\ 1 & 2 & 2^2 & \cdot & 2^{n-1} \\ 1 & 3 & 3^2 & \cdot & 3^{n-1} \\ \vdots & & \vdots & \vdots & \\ 1 & (n+2k) & (n+2k)^2 & \cdot & (n+2k)^{n-1} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-1} \end{bmatrix}$$

Berlekamp Welch

How do we find the original polynomial $P(x)$?

Suppose that $m_1, \ldots, m_k$ are the corrupted packets. Let $E(x) = (x - m_1) \ldots (x - m_k)$

Then $P(i) * E(i) = r_i * E(i)$ for any i greater than 1 and less than $n + 2k$. Why?

Solution: Case 1: i is corrupted

$$E(i) = 0 \rightarrow 0 = 0$$

Case 2: i is not corrupted

$$P(i) = r_i \rightarrow P(i) * E(i) = r_i E(i)$$

Let $Q(i) = P(i)E(i)$ So we have $Q(i) = P(i)E(i) = r_i * E(i)$ where $1 \leq i \leq 2k + n$ What degree is $Q(i)$?

Solution: $n + k - 1$

How many coefficients do we need to describe $Q(i)$?

Solution: $n + k$

What degree is $P(i)$?

Solution: k

How many unknown coefficients do we need to describe $E(i)$?

Solution: k (we know that the first coefficient has to be 1)

We can write $Q(i) = r_i E(i)$ for every$i$ that is $1 \leq i \leq 2k + n$.

How many equations do we have? How many unknowns?

Solution: $n + 2k$

Once we have the above described equations, how do we determine what $P(i)$ is?

Solution: Solve the equations to get the coefficients for $E(i)$ and $Q(i)$. Then divide $\frac{E(i)}{Q(i)}$ to get $P(i)$.

1.2 Questions

1. (a) Alice sends Bob a message of length 3 on the Galois Field of 5 (modular space of mod 5). Bob receives the following message: (3, 2, 1, 1, 1). Assuming that Alice is sending messages using the proper general error message sending scheme, set up the linear equations that, when solved, give you the $Q(x)$ and $E(x)$ needed to find the original $P(x)$.

Solution: Set up 5 equations for the five values of x that we have, such that

$$Q(x) = r_x * (x - b) \mod 5$$

where

$$Q(x) = a_3 * x_i^3 + a_2 * x_i^2 + a_1 * x_i + a_0$$

(r_i = ith received number)
In the form, for

$$x = x_i, a_3 * x_i^3 + a_2 * x_i^2 + a_1 * x_i + a_0 = r_i * (x - b) \mod 5$$

$$x = 1, (a_3 + a_2 + a_1 + a_0) = 3(1 - b) \mod 5$$
$$x = 2, (3a_3 + 4a_2 + 2a_1 + a_0) = 2(2 - b) \mod 5$$
$$x = 3, 2a_3 + 4a_2 + 3a_1 + a_0 = 1(3 - b) \mod 5$$
$$x = 4, 4a_3 + 1a_2 + 4a_1 + a_0 = 1(4 - b) \mod 5$$
$$x = 5, 0 + 0 + 0 + a_0 = 1(0 - b) \mod 5$$

After solving for these equations, you should get the following:

$$b = 3$$
$$a_3 = 1$$
$$a_2 = 3$$
$$a_1 = 3$$
$$a_0 = 2$$

(b) What is the encoded message that Alice actually sent? What was the original message? Which packet(s) were corrupted?

Solution: Now we have $P(x) = \frac{Q(x)}{E(x)}$. Plug in the coefficients for Q and E and divide (using polynomial long division) and you get $P(x) = x^2 + x + 1$. Using

the $P(x)$ that we found, we plug in the values 1, 2, 3, 4, 5 to find the encoded 5 packet message. $P(1) = 3, P(2) = 2, P(3) = 3, P(4) = 1, P(5) = 1$. The original message is the first 3: $P(1), P(2), P(3)$. Using the value of b, we know that the 3rd packet was corrupted, which we can confirm in the message that was received.

2 Secret Sharing

2.1 Questions

1. You want to send a super secret message consisting of 10 packets to the space station through some astronauts. You are afraid that some malicious spy people are going to tell the wrong message and make the space station go spiraling out of orbit. Assuming that up to 5 of the astronauts are malicious, design a scheme so that the group of astronauts (including the malicious ones) still find the correct message that you want to send. You can send any number of astronauts, but try to make the number that you have to send as small as possible. Astronauts can only carry one packet with them.

Solution: We can use the scheme provided in the notes for general errors. If there are up to 5 malicious astronauts, then we have up to 5 general errors in our transmission. To account for those errors and to send the original message we have to send at least the length of the original message plus the twice the possible number of general errors. Thus, we have to send 20 astronauts or more. Create a polynomial of degree 9 (which is the length of the message minus 1), such that $P(0), P(1), \ldots, P(10)$ is the original message, and give each of the 20 astronauts unique points on that polynomial. Then use the methods we used in the previous question to find the message.

3 Uncountability

3.1 Introduction

1. (a.) What does it mean for a set to be countably infinite?

Solution: There is a bijection between the set and the natural numbers (or any other countable set).

(b.) Do $\mathbb{N}$ and $\mathbb{Z}^+$ have the same cardinality? Does adding one element change cardinality?

> **Solution:** Do the hotel argument. Just take each person starting at some arbitrary point k, and slide them over by one. In other words, map everybody at some point $p \geq k$ to $p+1$, and then slide the newcomer into spot k.
>
> No, adding in one more point did not change the cardinality of the set.

(c.) Cantor-Bernstein Theorem: Suppose there is an injective function from set A to set B and there is an injective function from set B to set A. Then there is a bijection between A and B. Use this theorem to prove that Q is countable.

> **Solution:** This is used when proving that rational numbers are countable. We know that $|\mathbb{N}| \leq |Q|$ because every natural number is a rational number. Just need to show that $|Q| \leq |\mathbb{N}|$. Do the spiral proof (which is presented in the notes).

3.2 Questions

1. Are these sets countably infinite/uncountably infinite/finite? If finite, what is the order of the set?

   (a) Finite bit strings of length n.

   > **Solution:** Finite. There are 2 choices (0 or 1) for each bit, and n bits, so there are $2 \times 2 \times \ldots \times 2 = 2^n$ such bit strings.

   (b) All finite bit strings of length 1 to n.

   > **Solution:** Finite. By part (a), there are 2^1 bit strings of length 1, 2^2 of length 2, etc. Thus, there are $2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 2$.

   (c) All finite bit strings

   > **Solution:** Countably infinite. We can list these strings as follows: $\{0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, \ldots\}$. This gives us a bijection with the (countable) natural numbers, so these are countably infinite.

(d) All infinite bit strings

> **Solution:** Uncountably infinite. We can construct a bijection between this set and the set of real numbers between 0 and 1. We can represent these real numbers using binary e.g. they are of the form $0.0110001010110\ldots$. By diagonalization, the set of real numbers between 0 and 1 is uncountably infinite; therefore, so is this set.

(e) All finite or infinite bit strings.

> **Solution:** Uncountably infinite. This is the union of a countably infinite set (part c) and an uncountably infinite set (part d), so it is uncountably infinite.

2. Find a bijection between N and the set of all integers congruent to $1 \mod n$, for a fixed n.

> **Solution:** The set of integers congruent to 1 mod n is $A = \{1 + kn \mid k \in \mathbb{Z}\}$. Define $g : \mathbb{Z} \to A$ by $g(x) = 1 + x \times n$; this is a bijection because it is clearly one-to-one, and is onto by the definition of A. We can combine this with the bijective mapping $f : \mathbb{N} \to \mathbb{Z}$ from the notes, defined by $f(x) = \begin{cases} \frac{x}{2} & \text{if } x \text{ is even} \\ \frac{-(x+1)}{2} & \text{if } x \text{ is odd} \end{cases}$ Then $f \circ g$ is a function from $\mathbb{N}$ to A, which is a bijection.

3. **True/False**

(a) Every infinite subset of a countable set is countable

> **Solution:** True. Define set S as the infinite subset, and set T as the countable set. $S \subseteq T$, which means $|S| \leq |T|$. This means S is countable. S is also an infinite subset, so it is infinite. $\therefore S$ is countably infinite. numbers, there is a mapping between E and N.

(b) If A and B are both countable, then $A \times B$ is countable

> **Solution:** True. Can draw a bijection where first elem of A maps to 1, first elem of B maps to 2, second elem of A maps to 3, etc. This will include $A \times B$ at the end, and because there is a bijection from A to $\mathbb{N}$ and B to $\mathbb{N}$, there is a bijection from $A \times B$ to $\mathbb{N}$. There is clearly a mapping from $A \times B$ to $\mathbb{N} \times \mathbb{N}$, and $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$, so there is a bijective mapping from $A \times B$ to $\mathbb{N}$

(c) Every infinite set that contains an uncountable set is uncountable.

Solution: True. Let A be an uncountable subset of B. Assume that B is countable. $f : \mathbb{N} \to B$ is a bijection. There must be a subset M of $\mathbb{N}$ such that $f : M \to A$ is a bijection. Then A is countable. This is a contradiction. So B must be uncountable.

4 Self Reference

4.1 Introduction

The Halting Problem: Does a given program ever halt when executed on a given input?

$$\texttt{TestHalt(P,x)} = \begin{cases} \text{“yes”}, & \text{if program } P \text{ halts on input } x \\ \text{“no”}, & \text{if program } P \text{ loops on input } x \end{cases}$$

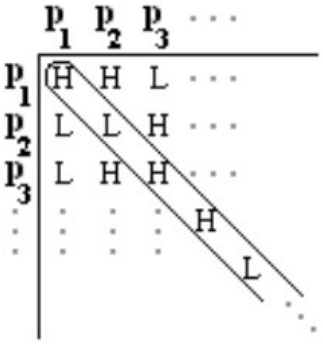
How do we prove that `TestHalt` does not exist? Lets assume that it does, and hope we reach a contradiction.
Define another program:

```
Turing(P)
  if TestHalt(P,P) = "yes" then loop forever
  else halt
```

What happens when we call `Turing(Turing)`?

Solution:
Case 1 : It halts. If `Turing(Turing)` halts then `TestHalt(Turing, Turing)` must have returned no. But `TestHalt(Turing Turing)` calls `Turing(Turing)` and calling `Turing(Turing)` must loop. But we assumed that `Turing(Turing)` halted. Contradiction.
Case 2 : It loops. This implies that `TestHalt(Turing, Turing)` returned yes, which by the way that `TestHalt` is defined implies that `Turing` halted. But we assumed that `Turing(Turing)` looped. Contradiction.

How is this just a reformulation of proof by diagonalization?



Solution: List all possible programs as rows and columns. The rows are the programs and the columns are the inputs. Turing must be one of the rows, say row n. If entry (n, n) is H then Turing will loop by definition. If entry (n, n) is L then Turing will halt by definition. Therefore Turing cannot be on the list of all programs and therefore it does not exist.

Therefore the Halting Problem is unsolvable. We can use this to prove that other problems are also unsolvable. Say we are asked if program M is solvable. To prove it is not, we just need to prove the following claim: If we can compute program M, then we could also compute the halting problem.
This would then prove that M can not exist, since the halting problem is not computable. This amounts to proof by contradiction.

4.2 Questions

1. Say that we have a program M that decides whether any input program halts as long as it prints out the string ABC as the first operation that it carries out. Can such a program exist? Prove your answer.

Solution: No. Such a program M can not exist. We proceed as follows: we show that if such a program existed, the halting problem would be computable.

Consider any program P. If we wanted to decide if P halted, we could simply create a new program P' where P' first prints out ABC, then proceed to do exactly what P would. However, if M existed, we could determine whether any program P halted by converting it to a P' and feeding it into M. This would solve the halting problem- but this is a contradiction, since by diagonalization we can prove that the halting problem is not solvable.

Therefore, M can not exist!

Below we have the Pseudocode for our program P.

```
define M'(Program p, input i):
    Construct program p' which will first print('ABC')
        then run p(i)
    run M(p', i)


define M(Program p, input i):
    if p's first command is print('ABC'):
        if p(i) halts:
            return True
        else if p(i) loops forever:
            return False
```

5 Intro to Counting

5.1 Introduction

Rules of counting:

1. If an object is made by a sequence of k choices, the number of ways to make the object is the number of ways to make the first choice, multiplied by the number of ways to make the second choice, and so on.
2. If the order does not matter, then count the number of ways to arrange the situation with order and then divide by the number of orderings/sorted objects.

5.2 When Order Matters

1. (a) You have 15 chairs in a room and there are 9 people. How many different ways can everyone sit down?

   Solution: $\frac{15!}{6!}$. There are 15 places to put the first person, then 14 places to put the second person, 13 places to put the third person, etc all the way to the last person who has 7 places to sit. Another way to think about this is like the anagram example above. We have 9 unique letters and 6 repeats (our empty spaces). We divide by the repeats giving us: $15*14*13*12*11*10*9*8*7 = \frac{15!}{6!}$

   (b) How many ways are there to fill 9 of the 15 chairs? (We dont care who sits in them)

   Solution: $\binom{15}{9} = \frac{15!}{9!(15-9)!}$ In this example, we don't care about the uniqueness of each person, so we can just count each person as a repeat. So like the anagram example we'll divide for every repeat. We have 9 human repeats, and 6 empty space repeats. Hence $\frac{15!}{9!6!}$

2. **Identical Digits** The numbers 1447, 1005, and 1231 have something in common. Each of them is a four digit number that begins with 1 and has two identical digits. How many numbers like this are there?

   Solution: Case 1: the identical digits are 1 (e.g. $11xy, 1x1y, 1xy1$)
   Since there can only be two numbers that are identical, x and y cannot be 1 and $x \neq y$.
   So [Possible formats] * [Possible x values] * [Possible y values] $= 3 * 9 * 8 = 216$

Case 2: identical digits are not 1 (e.g. $1xxy, 1xyx, 1yxx$).
So [Possible formats] * [Possible x values] * [Possible y values] $= 3 * 9 * 8 = 216$
Add both cases to arrive at the final result. $216 + 216 = 432$

5.3 More Practice

1. At Starbucks, you can choose either a Tall, a Grande, or a Venti drink. Further, you can choose whether you want an extra shot of espresso or not. Furthermore, you can choose whether you want a Latte, a Cappuccino, an Americano, or a Frappuccino.

   How many different drink combinations can you order?

   Solution: $3 \cdot 2 \cdot 4$ (# sizes * espresso or not * # types of coffee)

2. Let's grab a deck of cards – it's poker time! Remember, in poker, order doesnt matter. By rank, we refer to the face value of cards (i.e. the number or K/Q/J/A), not the suit.

   (a) How many ways can we have a hand with exactly one pair? This means a hand with ranks (a, a, b, c, d).

   Solution: $13 * \binom{4}{2} * \binom{12}{3} * 4^3$. There are 13 value options for a (2, 3, 4, . . . , K, A). We then need to choose 2 out of the 4 possible suits. Now we need to choose b, c, and d. There are 12 values left (must be different from a). Finally, there are 4 suit options for each of the values chosen for b, c, d.

   (b) How many ways can we have a hand with four of a kind? This means a hand with ranks (a, a, a, a, b).

   Solution: $13 * 12 * 4$

   (c) How many ways can we have a straight? A straight is 5 consecutive cards.

   Solution: A straight can begin from any number from 2-10: (2, 3, 4, 5, 6); (3, 4, 5, 6, 7); ; (10, J, Q, K, A). That gives us 9 possibilities. Each number in hand has 4 possibilities (suits), so we have $9 * (4^5)$.

   (d) How many ways can we have a hand of all of the same suit?

Solution: $4 * \binom{13}{5}$. For each of the 4 suits, there are $\binom{13}{5}$ different combinations of 5 cards among 13 to choose from.

(e) How many ways can we have a straight flush? This means we have a consecutive-rank hand of the same suit. For example, $(2, 3, 4, 5, 6)$, all of spades is a straight flush, while $(2, 3, 5, 7, 8)$ of all spades is NOT, as the ranks are not consecutive.

Solution: For each of 4 suits, there are 9 number combinations (as shown in c, starting from 2 to starting from 10). Each number combination is unique, because there is only one number per suit. Thus, the answer is $4 * 9 = 36$.