POLYNOMIALS, SECRET SHARING, ERASURE ERRORS, GENERAL ERRORS, SELF REFERENCE

COMPUTER SCIENCE MENTORS 70

October 3 to October 7, 2016

1 Polynomials

1.1 Introduction

1. There is a unique polynomial of degree $n-1$ such that $P(i) = m_i$ for each packet $m_1, \ldots, m_n$
2. To account for errors we send $c_1 = P(1), \ldots, c_{n+j} = P(n+j)$
3. If polynomial $P(x)$ has degree $n-1$ then we can uniquely reconstruct it from any n distinct points.
4. If a polynomial $P(x)$ has degree $n-1$ then it can be uniquely described by its n coefficients

1.2 Questions

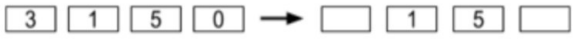
1. Define the sequence of polynomials by $P_0(x) = x+12$, $P_1(x) = x^2-5x+5$ and $P_n(x) = xP_{n-2}(x) - P_{n-1}(x)$. (For instance, $P_2(x) = 17x - 5$ and $P_3(x) = x^3 - 5x^2 - 12x + 5$.)
   (a) Show that $P_n(7) \equiv 0 \pmod{19}$ for every $n \in N$.

(b) Show that, for every prime q, if $P_{2013}(x) \not\equiv 0 \pmod{q}$, then $P_{2013}(x)$ has at most 2013 roots modulo q.

2 Erasure Errors

2.1 Introduction

We want to send n packets and we know that k packets could get lost.



How many more points does Alice need to send to account for k possible errors? __
What degree will the resulting polynomial be? ___
How large should q be if Alice is sending n packets with k erasure errors, where each packet has b bits?

What would happen if Alice instead send $n+k-1$? Why will Bob be unable to recover the message?

2.2 Questions

1. Suppose $A = 1$, $B = 2$, $C = 3$, $D = 4$, and $E = 5$. Assume we want to send a message of length 3. Recover the lost part of the message, or explain why it can not be done.

   1. C_AA

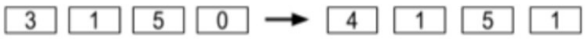
   2. CE_ _

2. Suppose we want to send n packets, and we know $p = 20\%$ of the packets will be erased. How many extra packets should we send? What happens if p increases (say to 90%)?

3 General Errors

3.1 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 $P(x)$.

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

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

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

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

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

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

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?

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 $Q(i)$?

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

What degree is $P(i)$?

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

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?

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

3.2 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 $P(x)$.

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

2. Suppose that the message we want to send consists of 10 numbers. We find a polynomial of degree 9 which goes through all these points and evaluate it on 25 points. How many erasure errors can we recover from? How about general errors?

3. You want to send a super secret message consisting of 10 packets to the space station through some astronauts. Youre 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.

4 Secret Sharing

4.1 Questions

1. Suppose the Oral Exam questions are created by 2 TAs and 3 Readers. The answers are all encrypted and we know that:
   (a) Both TAs should be able to access the answers
   (b) All 3 Readers can also access the answers
   (c) One TA and one Reader should also be able to do the same

   Design a secret sharing scheme to make this work.

2. An officer stored an important letter in her safe. In case she is killed in battle, she decides to share the password with her troops. Everyone knows there are 3 spies among the troops, but no one knows who they are except for the three spies themselves.The 3 spies can coordinate with each other and they will either lie and make people not able to open the safe, or will open the safe themselves if they can. Therefore, the officer would like a scheme to share the password that satisfies the following conditions:

   1. When M of them get together, they are guaranteed to be able to open the safe even if they have spies among them.
   2. The 3 spies must not be able to open the safe all by themselves.

   Please help the officer to design a scheme to share her password. What is the scheme? What is the smallest M? Show your work and argue why your scheme works and any smaller M couldnt work.

5 Self Reference

5.1 Introduction

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

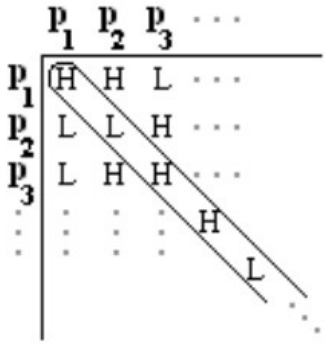
$$\texttt{TestHalt(P,x)} = \begin{cases} \text{“yes”}, & \text{if program } P \text{ halts on input } x \\ \text{“no”}, & \text{if program } P \text{ loops on input } x \end{cases}$$

How do we prove that `TestHalt` doesnt 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)`?

How is this just a reformulation of proof by diagonalization?



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.

5.2 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?