

15-251

Great Theoretical Ideas in Computer Science

W

Polynomials, Lagrange, and Error-correction

Lecture 16 (March 5, 2009)

$$P(X) = \text{stick figure} X^3 + \text{stick figure} X^2 + \text{stick figure} X^1 + \text{stick figure}$$

Fields

Definition:

A field F is a set together with two binary operations $+$ and \times , satisfying the following properties:

1. $(F, +)$ is a commutative group
2. $(F - \{0\}, \times)$ is a commutative group
3. The distributive law holds in F :
 $(a + b) \times c = (a \times c) + (b \times c)$

Polynomials in one variable over the reals

$$P(x) = 3x^2 + 7x - 2$$

$$Q(x) = x^{123} - \frac{1}{2}x^{25} + 19x^3 - 1$$

$$R(y) = 2y + \sqrt{2}$$

$$S(z) = z^2 - z - 1$$

$$T(x) = 0$$

$$W(x) = \pi$$

Representing a polynomial

A degree- d polynomial is represented by its $(d+1)$ coefficients:

$$P(x) = c_d x^d + c_{d-1} x^{d-1} + \dots + c_1 x^1 + c_0$$

The $d+1$ numbers c_d, c_{d-1}, \dots, c_0 are coefficients.

E.g. $P(x) = 3x^4 - 7x^2 + 12x - 19$

Coefficients are:

Are we working over the reals?

We could work over any field
(set with addition, multiplication, division defined.)

E.g., we could work with the rationals, or the reals.

Or with $(\mathbb{Z}_p, +, \times)$, the integers mod prime p .

In this lecture, we will work with \mathbb{Z}_p

the field $(\mathbb{Z}_p, +, \cdot)$

$(\mathbb{Z}_p = \{0, 1, 2, \dots, p-1\}, +)$
is a commutative group

$(\mathbb{Z}_p^\times = \{1, 2, 3, \dots, p-1\} = \mathbb{Z}_p \setminus \{0\}, \cdot)$
is also a commutative group

$(\mathbb{Z}_p, +, \cdot)$
is a field

Addition distributes over multiplication.

Let $\mathbb{Z}_p[x]$ denote the set of polynomials with variable x and coefficients from \mathbb{Z}_p

Multiplying Polynomials

(say from $\mathbb{Z}_{11}[x]$)

$$(x^2+2x-1)(3x^3+7x)$$

$$= x^2(3x^3 + 7x) + 2x(3x^3 + 7x) - (3x^3 + 7x)$$

$$= 3x^5 + 6x^4 + 4x^3 + 14x^2 - 7x$$

$$= 3x^5 + 6x^4 + 4x^3 + 3x^2 + 4x$$

Adding, Multiplying Polynomials

Let $P(x), Q(x)$ be two polynomials.

The sum $P(x)+Q(x)$ is also a polynomial.

(i.e., polynomials are "closed under addition")

Their product $P(x)Q(x)$ is also a polynomial.

("closed under multiplication")

$P(x)/Q(x)$ is not necessarily a polynomial.

$\mathbb{Z}_p[x]$ is a commutative ring with identity

Let $P(x), Q(x)$ be two polynomials.

The sum $P(x)+Q(x)$ is also a polynomial.

(i.e., polynomials are "closed under addition")

Addition is associative

0 (the "zero" polynomial) is the additive identity

$-P(x)$ is the additive inverse of $P(x)$

Also, addition is commutative

$(\mathbb{Z}_p[x], +)$ is a commutative group

$\mathbb{Z}_p[x]$ is a commutative ring with identity

Let $P(x), Q(x)$ be two polynomials.

The sum $P(x) \cdot Q(x)$ is also a polynomial.

(i.e., polynomials are "closed under multiplication")

Multiplication is associative

1 (the "unit" polynomial) is the multiplicative identity

Multiplication is commutative

Finally, addition distributes over multiplication

$(\mathbb{Z}_p[x], +, \cdot)$ is a commutative ring with identity

(mult. inverses may not exist)

Evaluating a polynomial

Suppose:

$$P(x) = c_d x^d + c_{d-1} x^{d-1} + \dots + c_1 x^1 + c_0$$

E.g. $P(x) = 3x^4 - 7x^2 + 12x - 19$

$$P(5) = 3 \times 5^4 - 7 \times 5^2 + 12 \times 5 - 19$$

$$P(-1) = 3 \times (-1)^4 - 7 \times (-1)^2 + 12 \times (-1) - 19$$

$$P(0) = -19$$

The roots of a polynomial

Suppose:

$$P(x) = c_d x^d + c_{d-1} x^{d-1} + \dots + c_1 x^1 + c_0$$

Definition: r is a "root" of $P(x)$ if $P(r) = 0$

E.g., $P(x) = 3x + 7$ root = $-(7/3)$.

$P(x) = x^2 - 2x + 1$ roots = 1, 1

$P(x) = 3x^3 - 10x^2 + 10x - 2$ roots = $1/3, 1, 2$.

Linear Polynomials

$$P(x) = ax + b$$

$$\text{One root: } P(x) = ax + b = 0 \quad \Rightarrow x = -b/a$$

E.g., $P(x) = 7x - 9$ in $\mathbb{Z}_{11}[x]$

$$\text{root} = (-(-9)/7) = 9 * 7^{-1}$$

$$= 9 * 8 = 72$$

$$= 6 \pmod{11}.$$

$$\text{Check: } P(6) = 7*6 - 9 = 42 - 9 = 33 = 0 \pmod{11}$$

The Single Most Important Theorem About Low-degree Polynomials

**A non-zero degree- d
polynomial $P(x)$ has
at most d roots.**

This fact has many applications...

An application: Theorem

Given pairs $(a_1, b_1), \dots, (a_{d+1}, b_{d+1})$ of values
there is at most one
degree- d polynomial $P(x)$
such that:
 $P(a_k) = b_k$ for all k

when we say "degree- d ", we mean
degree at most d .

we'll always assume $a_i \neq a_k$ for $i \neq k$

An application: Theorem

Given pairs $(a_1, b_1), \dots, (a_{d+1}, b_{d+1})$ of values
there is at most one
degree- d polynomial $P(x)$
such that:

$$P(a_k) = b_k \text{ for all } k$$

Let's prove the contrapositive

Assume $P(x)$ and $Q(x)$ have degree at most d

Suppose a_1, a_2, \dots, a_{d+1} are $d+1$ points
such that $P(a_k) = Q(a_k)$ for all $k = 1, 2, \dots, d+1$

Then $P(x) = Q(x)$ for all values of x

Proof: Define $R(x) = P(x) - Q(x)$

$R(x)$ has degree d (at most)

$R(x)$ has $d+1$ roots (so it must be the zero
polynomial) (fund thm of algebra) \square

Theorem:

Given pairs $(a_1, b_1), \dots, (a_{d+1}, b_{d+1})$ of values
there is at most one
degree-d polynomial $P(x)$
such that:
 $P(a_k) = b_k$ for all k

do there exist $d+1$ pairs
for which there are
no such polynomials??

Revised Theorem:

Given pairs $(a_1, b_1), \dots, (a_{d+1}, b_{d+1})$ of values
there is exactly one
degree-d polynomial $P(x)$
such that:
 $P(a_k) = b_k$ for all k



The algorithm to construct $P(x)$
is called Lagrange Interpolation

Two different representations

$P(x) = c_d x^d + c_{d-1} x^{d-1} + \dots + c_1 x^1 + c_0$
can be represented either by

- its $d+1$ coefficients
 $c_d, c_{d-1}, \dots, c_2, c_1, c_0$
- Its value at any $d+1$ points
 $P(a_1), P(a_2), \dots, P(a_d), P(a_{d+1})$
(e.g., $P(1), P(2), \dots, P(d+1)$.)

Converting Between The Two Representations

Coefficients to Evaluation:

Evaluate $P(x)$ at $d+1$ points

Evaluation to Coefficients:

Use Lagrange Interpolation

Now for some Lagrange Interpolation

Given pairs $(a_1, b_1), \dots, (a_{d+1}, b_{d+1})$ of values
there is exactly one
degree-d polynomial $P(x)$
such that:
 $P(a_k) = b_k$ for all k

Special case

What if the points were like:

$(a_1, 1)$ ← b_1
 $(a_2, 0)$ ← b_2
 $(a_3, 0)$
...
 $(a_{d+1}, 0)$ ← b_{d+1}

Special case

(a₁, b₁) (a₂, b₂) ... (a_{d+1}, b_{d+1})

Suppose we can get degree-d poly $h_1(x)$:

$h_1(a_1) = 1$
 $h_1(a_t) = 0$ for all $t = 2, \dots, d+1$

“switch” polynomial #1

Special case

Suppose we can get degree-d poly $h_1(x)$:

$h_1(a_1) = 1$
 $h_1(a_t) = 0$ for all $t = 2, \dots, d+1$

Then we can get degree-d poly $H_1(x)$:

$H_1(a_1) = b_1$
 $H_1(a_t) = 0$ for all $t = 2, \dots, d+1$

Just set $H_1(x) = b_1 * h_1(x)$

Special case

Suppose we can get degree-d poly $h_1(x)$:

$h_1(a_1) = 1$
 $h_1(a_t) = 0$ for all $t = 2, \dots, d+1$

Using same idea, get degree-d poly $H_k(x)$:

$H_k(a_k) = b_k$
 $H_k(a_t) = 0$ for all $t \neq k$

*P(a₁) = b₁
P(a₂) = b₂ ...*

Finally, $P(x) = \sum_k H_k(x)$

Hence, all we need to do

Given numbers a_1, a_2, \dots, a_{d+1}

Build a degree-d “switch” poly $h_1(x)$:

$h_1(a_1) = 1$
 $h_1(a_t) = 0$ for all $t = 2, \dots, d+1$

construction by example

want a quadratic h with $h(3) = 1, h(1) = 0, h(6) = 0$
(say, in $\mathbb{Z}_{11}[x]$)

Let's first get the roots in place:

$h(x) = (x-1)(x-6)$

Are we done? No! We wanted $h(3) = 1$

But $h(3) = (3-1)(3-6) = -6$ So let's fix that!

$h(x) = (-6)^{-1} (x-1)(x-6)$
 $= 9 (x-1)(x-6)$ done!

9 * (-6) = -54 = 11

formally, the constructions

k-th "Switch" polynomial

$$g_k(x) = (x-a_1)(x-a_2)\dots(x-a_{k-1})(x-a_{k+1})\dots(x-a_{d+1})$$

Degree of $g_k(x)$ is: d

$g_k(x)$ has d roots: $a_1, \dots, a_{k-1}, a_{k+1}, \dots, a_{d+1}$

$$g_k(a_k) = (a_k-a_1)(a_k-a_2)\dots(a_k-a_{k-1})(a_k-a_{k+1})\dots(a_k-a_{d+1})$$

For all $i \neq k$, $g_k(a_i) = 0$

k-th "Switch" polynomial

$$g_k(x) = (x-a_1)(x-a_2)\dots(x-a_{k-1})(x-a_{k+1})\dots(x-a_{d+1})$$

$$h_k(x) = \frac{(x-a_1)(x-a_2)\dots(x-a_{k-1})(x-a_{k+1})\dots(x-a_{d+1})}{(a_k-a_1)(a_k-a_2)\dots(a_k-a_{k-1})(a_k-a_{k+1})\dots(a_k-a_{d+1})}$$

$$h_k(a_k) = 1$$

For all $i \neq k$, $h_k(a_i) = 0$

The Lagrange Polynomial

$$h_k(x) = \frac{(x-a_1)(x-a_2)\dots(x-a_{k-1})(x-a_{k+1})\dots(x-a_{d+1})}{(a_k-a_1)(a_k-a_2)\dots(a_k-a_{k-1})(a_k-a_{k+1})\dots(a_k-a_{d+1})}$$

$$P(x) = b_1 h_1(x) + b_2 h_2(x) + \dots + b_{d+1} h_{d+1}(x)$$

$P(x)$ is the unique polynomial of degree d such that $P(a_1) = b_1$, $P(a_2) = b_2$, ..., $P(a_{d+1}) = b_{d+1}$

Example

Input: $(5,1)$, $(6,2)$, $(7,9)$ Want quadratic in $Z_{11}[x]$

Switch polynomials:

$$h_1(x) = (x-6)(x-7)/(5-6)(5-7) = \frac{1}{2} (x-6)(x-7)$$

$$h_2(x) = (x-5)(x-7)/(6-5)(6-7) = -(x-5)(x-7)$$

$$h_3(x) = (x-5)(x-6)/(7-5)(7-6) = \frac{1}{2} (x-5)(x-6)$$

$$\begin{aligned} P(x) &= 1 \times h_1(x) + 2 \times h_2(x) + 9 \times h_3(x) \\ &= (3x^2 - 32x + 86) \\ &= (3x^2 + x + 9) \text{ in } Z_{11}[x] \end{aligned}$$

the Chinese Remainder Theorem
uses very similar ideas in
its proof

Revised Theorem:

Given pairs $(a_1, b_1), \dots, (a_{d+1}, b_{d+1})$ of values
there is exactly one
degree- d polynomial $P(x)$
such that:
 $P(a_k) = b_k$ for all k



The algorithm to construct $P(x)$
is called Lagrange Interpolation

An Application:
Error Correcting Codes

Error Correcting Codes

Messages as sequences of numbers in Z_{29} :

HELLO 8 5 12 12 15

I want to send a sequence of $d+1$ numbers
Suppose your mailer may corrupt any k
among all the numbers I send.
How should I send the numbers to you?

In Particular

Suppose I just send over the numbers
8 5 12 12 15 say $k=2$ errors
and you get
8 9 0 12 15 *Hi LO*

How do you correct errors?
How do you even detect errors?

A Simpler Case: Erasures

Suppose I just send over the numbers
8 5 12 12 15 say $k=2$ erasures
and you get
8 * * 12 15

(Numbers are either correct or changed to *)
What can you do to correct errors?

A Simple Solution

Repetition: repeat each number $k+1$ times

8 8 8 5 5 5 12 12 12 12 12 12 15 15 15

At least one copy of each number will reach

8 8 8 5^(**) 12 12 12 12 12 12 15 15 15
5 5 5 6

For arbitrary corruptions, repeat $2k+1$ times
and take majority

Very wasteful!

To send $d+1$ numbers with erasures, we sent
 $(d+1)(k+1)$ numbers

Can we do better?

Note that to send 1 number with k erasures
we need to send $k+1$ numbers.

Think polynomials...

Encoding messages as polynomials:

HELLO 8 5 12 12 15

$$\underline{8}x^4 + \underline{5}x^3 + \underline{12}x^2 + 12x + 15 \in \mathbb{Z}_{29}[x]$$

I want to send you a polynomial P(x) of degree d.

Send it in the value representation!

Want to send a polynomial of degree-d subject to at most k erasures.

Evaluate P(x) at d+1+k points *caution (d+k) (k n)*

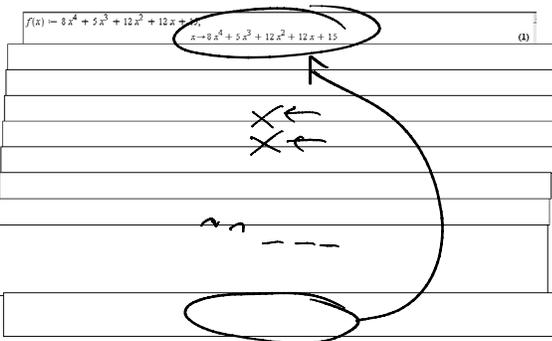
Send P(0), P(1), P(2), ..., P(d+k)

At least d+1 of these values will reach

Say P(0), *, P(2), *, ..., *, P(d+k)

Can recover P(x) from these d+1 values

Example



Example

```
f(x) = 8x^4 + 5x^3 + 12x^2 + 12x + 15
x -> 0x^4 + 5x^3 + 12x^2 + 12x + 15 (1)
f(0) mod 29 15 (2)
f(1) mod 29 23 (3)
f(2) mod 29 23 (4)
f(3) mod 29 14 (5)
f(4) mod 29 13 (6)
f(5) mod 29 26 (7)
f(6) mod 29 19 (8)
CurveFitting[PolynomialInterpolation][{10, 15}, {1, 23}, {4, 13}, {5, 26}, {6, 19}], x, form = Lagrange);
1/8 (x - 1) (x - 4) (x - 5) (x - 6) - 23/80 x (x - 4) (x - 5) (x - 6) + 13/24 x (x - 1) (x - 5) (x - 6) - 13/10 x (x - 1) (x - 4) (x - 6) + 19/80 x^2 (x - 1) (x - 4) (x - 5)
g := eval[expand(%), mod 29]
8x^4 + 5x^3 + 12x^2 + 12x + 15 (10)
```

Much better!!!

Naïve Repetition:

To send d+1 numbers with k erasures, we sent (d+1)(k+1) numbers

Polynomial Coding:

To send d+1 numbers with k erasures, we sent (d+k+1) numbers

What about corruptions?

Want to send a polynomial of degree-d subject to at most k corruptions.

Suppose we try the same idea

Evaluate P(x) at d+1+k points

Send P(0), P(1), P(2), ..., P(d+k)

At least d+1 of these values will be unchanged

Example

$P(x) = 2x^2 + 1$, and $k = 1$.

So I sent $P(0)=1$, $P(1)=3$, $P(2)=9$, $P(3)=19$

Corrupted email says (1, 4, 9, 19)

Choosing (1, 4, 9) will give us $Q(x) = x^2 + 2x + 1$

Choosing (1, 9, 19) gives us back $P(x)$

But we can at least detect errors!

Evaluate $P(x)$ at $d+1+k$ points

Send $P(0), P(1), P(2), \dots, P(d+k)$

At least $d+1$ of these values will be correct

Say $P(0), P'(1), P(2), P(3), P'(4), \dots, P(d+k)$

Using these $d+1$ correct values will give $P(x)$

Using any of the incorrect values will give something else

Quick way of detecting errors

Interpolate first $d+1$ points to get $Q(x)$

Check that all other received values are consistent with this polynomial $Q(x)$

If all values consistent, no errors!

In that case, we know $Q(x) = P(x)$

else there were errors...

Number of numbers?

Naïve Repetition:

To send $d+1$ numbers with error detection,
sent $(d+1)(k+1)$ numbers

Polynomial Coding:

To send $d+1$ numbers with error detection,
sent $(d+k+1)$ numbers

How about error correction?

requires more work

To send $d+1$ numbers in such a way
that we can correct up to k errors,
need to send $d+1+2k$ numbers.

Similar encoding scheme

Evaluate degree- d $P(x)$ at $d+1+2k$ points

Send $P(0), P(1), P(2), \dots, P(d+2k)$

At least $d+1+k$ of these values will be correct

Say $P(0), P(1), P(2), P(3), P(4), \dots, P(d+2k)$

How do we know which are correct?

how do we do this fast?

Theorem: A unique degree- d polynomial $R(x)$ can agree with the received data on at least $d+1+k$ points

Clearly, the original polynomial $P(x)$ agrees with data on $d+1+k$ points (since at most k errors, total $d+1+2k$ points)

And if two different degree- d polynomials did so, they would have to agree with each other on $d+1$ points, and hence be the same.

So any such $R(x) = P(x)$

Theorem: A unique degree- d polynomial $R(x)$ can agree with the received data on at least $d+1+k$ points

Brute-force Algorithm:

Interpolate each subset of $(d+1)$ points

Check if the resulting polynomial agrees with received data on $d+1+k$ pts

$\binom{d+1+k}{d+1}$

Takes too much time...

A fast algorithm to decode was given by Berlekamp and Welch which solves a system of linear equations

Recent research has given very fast encoding and decoding algorithms

BTW, this coding scheme is called Reed-Solomon encoding

It's used in practice...

FR: TELECOM SUPPLIER
150 HOUGHTON DRIVE
W LONG BEACH, NJ
07764-1294

TO: TELECOM CUSTOMER
2820 VALLEYDALE ROAD
BIRMINGHAM, AL
35244

CARD

(S) REC ID: 0662742MV96421234

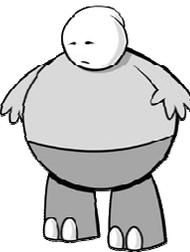
(B) CUSTOMER ID: AA00211211 RINGER C4C

(Q) QUANTITY: 1 EA PACKAGE COUNT: 1 OF 1
PACKAGE WEIGHT: 3 LBS

CUST

Maxicodes
= "UPS codes"
= another 2-d
Reed-Solomon codes

PDF417 codes
= 2-d Reed-Solomon
codes



Polynomials

Fundamental Theorem of polys:
Degree- d poly has at most d roots.
Two deg- d polys agree on $\leq d$ points.

Lagrange Interpolation:
Given $d+1$ pairs (a_k, b_k) , can find unique poly P with $P(a_k) = b_k$ for all these k .
Gives us value represent'n for polys.

Here's What You Need to Know...

- Error Correction
- Erasures codes
- Detection and correction