A Theorist's Toolkit **(CMU 18-859T, Fall 2013)**

Lecture 9: Fields and Polynomials

October 7th, 2013

Lecturer: Ryan O'Donnell *Scribe: Kevin Su*

1 Introduction

Moving on from spectral graph theory, this lecture will cover fields and polynomials. As a quick refresher, a field is a number system that includes the operations $\{+, -, \times, \div\}$. Recall that a ring is a number system with just $\{+, -, \times\}$. Fields can be infinite, such as $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$. We'll concern ourselves with finite fields in this lecture.

For polynomials, we'll talk about both univariate and multivariate polynomials. A polynomial for example might be $c_1x_1^3 + c_2x_2x_3^5$. Note that the coefficients here will come from a field $\mathbb{F}$. Then we write the set of polynomials as $\mathbb{F}[x_1, x_2, x_3]$.

As a quick roadmap of where we'll be going this lecture, here is a **TL; DR:**

1) If p is a prime, then $\{x \bmod p \mid x \in \mathbb{Z}\}$ is a field, denoted $\mathbb{F}_p$

2) For all $l \in \mathbb{N}$, there also exists a field with p^l elements, denoted $\mathbb{F}_{p^l}$

3) A degree d, non-zero univariate polynomial has at most d roots

4) Let $\mathrm{P} \in \mathbb{F}_q[x_1, \ldots, x_n]$ be a non-zero polynomial with *total degree* $\leq d$. Then:

$$\mathop{\mathbf{Pr}}_{a_1,\ldots,a_n \sim \mathbb{F}_q}\Big[\mathrm{P}(a_1, \ldots, a_n) = 0\Big] \leq \frac{d}{q}$$

2 Fields

We'll take for granted that $\mathbb{Z}_p$ is a ring, where $\mathbb{Z}_p$ is defined to be normal arithmetic modulo p. To show that $\mathbb{Z}_p$ is a field, we need to find a multiplicative inverse for each element. That is, given $a \in \mathbb{Z}_p \setminus \{0\}$, we want to find a^{-1}. Instead of just showing that a^{-1} exists, we'll actually give a construction for finding a^{-1}.

The basic idea is to use the Extended Euclidean Algorithm. Recall that the E.E.A. finds the gcd of two numbers (we could in fact use any algorithm that finds the gcd). Given two numbers a, b, and if $a > b$, it divides the larger by the smaller, which gives $a = bq + r$, and uses the pair b, r at

the next step, until $r = 0$. For example, we might try $(100, 45)$:

$$\begin{aligned}(100, 45) &\implies 100 = 45 \cdot 2 + 10\\ (45, 10) &\implies 45 = 10 \cdot 4 + 5\\ (10, 5) &\implies 10 = 5 \cdot 2 + 0\\ (5, 0) &\implies \gcd(100, 45) = 5\end{aligned}$$

If we're smart with the bookkeeping of the algorithm, we can generate $c, d \in \mathbb{Z}$ such that:

$$\begin{aligned}&c \cdot a + d \cdot p = \gcd(a, p)\\ &c \cdot a + d \cdot p = 1 && \text{since } \gcd(a, p) = 1\\ &c \cdot a + d \cdot 0 \equiv 1 \mod p && \text{since } p \equiv 0 \mod p\\ &\quad c \cdot a \equiv 1 \mod p\\ &\implies c \equiv a^{-1} \mod p\end{aligned}$$

Note we can do this efficiently in polylog(p) time, and therefore we have shown that $\mathbb{Z}_p$ is a field, which we write as $\mathbb{F}_p$.

Aside: It is good to remember the fact that for $a \in \mathbb{F}_p, n \in \mathbb{N}$, we can compute $a^n \in \mathbb{F}_p$ in poly$(\log p, \log n)$ time (we can square a, then square the result, and so forth, taking each result mod p).

3 Picking a prime

In this section, we explore the question: How do you pick a prime? We know that picking a prime gives us a field, so this is an important question to answer. In general, we can use the (not so) elegant method below:

1) Pick a random m-bit number
2) Run a primality test on the selected number. If prime, output p. Else, go to **1**.

An important point is that testing a number for primality can be done in deterministic poly(m) time with the AKS primality test [AKS04].

A natural question to ask is why don't we just start at some 2^m, test if it's prime, then try $2^m + 1$, $2^m + 2$, and so forth. There are a couple reasons. First, in some cases, we actually want a random prime number (for example cryptography). Secondly, there can actually be long stretches where there are no prime numbers. Fortunately, the prime number theorem (of which you proved a weaker version in homework 1) states:

$$\text{fraction of } m\text{-bit numbers which are prime} \sim \frac{1}{2\ln(2) \cdot m}$$

This is good news for us! Recall that if a weighted coin is heads with probability h, then the expected number of flips until we see a heads is $\frac{1}{h}$. Therefore, the expected number of trials we

have to run until we find a prime is $\sim 2\ln(2) \cdot m \in O(m)$.

4 Other fields

One note: there are also finite fields not of this form. Specifically, there are finite fields of size 2^l which are particularly useful for computer scientists. An example of another field is:

$$\left\{a + bi : a, b \in \mathbb{F}_3\right\}$$

where $i = \sqrt{-1}$. This field is actually (isomorphic to) $\mathbb{F}_9$. Another interesting point: if a, b are drawn from $\mathbb{F}_5$, then this is *not* a field.

5 Randomized algorithms

Before we move onto polynomials, we'll take a quick detour to discuss randomized algorithms. This will help later in the course when we cover derandomization. We define several classes of randomized algorithms:

Definition 5.1. Zero-sided error: These algorithms run in expected poly time, and the answer is always correct (note the answer might be "I don't know").

Definition 5.2. One-sided error: These algorithms can make an error on one side with small probability. For example, the Miller-Rabin test for primality always outputs "prime" if the number is prime, but may occasionally output "prime" even if the number is composite (although with very low probability).

Definition 5.3. Two-sided error: These algorithms may output wrong answers for both cases, but with very low probability.

You might have heard the term "Las Vegas algorithms" for zero-sided error and "Monte Carlo algorithms" for one-sided error. While prevalent in literature, these make less sense as names (Ryan suggests that it's possible that the name Las Vegas algorithms comes from the fact that the house *always* wins, but who knows).

Note in practice, we would actually use the Miller-Rabin test, since it's roughly $O(m^3)$, whereas the AKS primality test is about $O(m^{7.5})$.

6 Polynomials

We'll also want to discuss a topic related to fields. We write a set of polynomials where the coefficients are taken from the field $\mathbb{F}$ as $\mathbb{F}[x_1, \ldots, x_n]$. A quick note on degree:

Definition 6.1. Total degree: max degree of any monomial (term).
Definition 6.2. Individual degree of x_i: max degree of x_i.

For example, given the polynomial $c_1 x_1^3 x_2^4 + c_2 x_1^2 x_3 x_5^2$, it has total degree 7, and the individual degree of $x_1 = 3$.

Before we move on, we make a quick note of a fact.

Fact: We can do univariate division with a remainder. Given $\mathrm{A}(x), \mathrm{B}(x)$, we can write

$$\mathrm{B}(x) = \mathrm{A}(x) \cdot \mathrm{Q}(x) + \mathrm{R}(x)$$

which corresponds to dividing $\mathrm{B}(x)$ by $\mathrm{A}(x)$. Additionally, here $\deg(\mathrm{R}) < \deg(\mathrm{A})$.

Corollary 6.3. *There is an analog to the Euclidean Algorithm for two univariate polynomials* $\mathrm{A}(x)$ *and* $\mathrm{B}(x)$*. The output is the highest degree polynomial which evenly divides both* $\mathrm{A}(x)$ *and* $\mathrm{B}(x)$*.*

Given this corollary, we see that there is a "gcd" between two polynomials. We can now define an irreducible polynomial.

Definition 6.4. A polynomial $\mathrm{P}(x)$ with $\deg(\mathrm{P}) > 1$ is *irreducible* (*prime*) if and only if $\gcd(\mathrm{P}(x), \mathrm{Q}(x)) = e$, where $e \in \mathbb{F}$, for any $\mathrm{Q}(x)$ where $\deg(\mathrm{Q}) < \deg(\mathrm{P})$.

Equivalently, a polynomial is irreducible if it can't be factored.

7 Fields with polynomials

We can now get other fields by taking polynomials modulo an irreducible polynomial. Note that in general, polynomials defined over a field form a ring. By taking everything modulo an irreducible, we can define a field as follows:

$$\left\{ \mathbb{F}[x] \bmod \mathrm{P}(x) \right\} \text{ is a field of size } |\mathbb{F}|^{\deg(\mathrm{P})}$$

Note the modulo here works similar to integers. For example, any polynomial mod $x^3 + x^2 + \ldots$ will yield a polynomial of at most degree 2 (i.e. $x^2 + x$). One can also think of an element as a list of coefficients in $\mathbb{F}$. Given $\mathbb{F}_p$ and an irreducible polynomial of degree l, we can do arithmetic in $\mathbb{F}_p^l$ in time $\mathrm{poly}(l \cdot \log p)$. If we write $q = p^l$, then this is just $\mathrm{poly}(\log q)$, where $\log q$ is the number of bits needed to write an element.

Given a general field element $c_0 + c_1 x + \ldots + c_{l-1}x^{l-1} \bmod \mathrm{P}(x)$, we can see that F_p^l is a vector space over $\mathbb{F}_p$. There are a couple more brief comments before we move on from finite fields.

For finding an irreducible polynomial, we use the same strategy as finding primes—pick one at random and test it. Of course, this only works if two things hold. First, we need to be able to efficiently check if a polynomial is irreducible. Secondly, irreducible polynomials must occur with a reasonable probability.

Fact: Checking if a polynomial $\mathrm{P}(x)$ of degree l is irreducible is in deterministic $\mathrm{poly}(l \log p)$ time.

Fact: If we pick $\mathrm{P}(x) = x^l + c_{l-1}x^{l-1} + \ldots + c_0$ randomly (where $c_i \in \mathbb{F}_p$),

$$\frac{1}{2l} < \mathbf{Pr}[\mathrm{P}(x) \text{ irreducible}] < \frac{1}{l}$$

And in fact, asymptotically is $\sim \frac{1}{l}$.

In some cases, we care about fields of size 2^l, and in this case we don't mind having factors of $\mathrm{poly}(p)$ in our runtime, since they are just $\mathrm{poly}(2)$. There are a couple results that are of interest to us then:

Theorem 7.1. *There exists a deterministic,* $\mathrm{poly}(p, l)$ *time algorithm to find a degree* l *irreducible polynomial modulo* p*. [Sho90]*

The next theorem is due to Van Lint, and defines a pattern of irreducible polynomials modulo 2.

Theorem 7.2. $x^2 + x + 1$*,* $x^6 + x^3 + 1$*,* $x^{18} + x^9 + 1$*, and in general* $x^{2\cdot(3k)} + x^{3k} + 1$ *are irreducible modulo 2 for all* $k \geq 0$*.*

For now, that's it for finite fields. The main takeaway is that fields of p^l exist, and you can do all arithmetic efficiently in it.

8 Roots of polynomials

We begin our study of roots of polynomials with an elementary fact:
Theorem 8.1. *Degree Mantra: A non-zero, univariate polynomial with degree* d *has at most* d *roots.*

Proof. Given $\mathrm{P}(x)$, write it for example as:

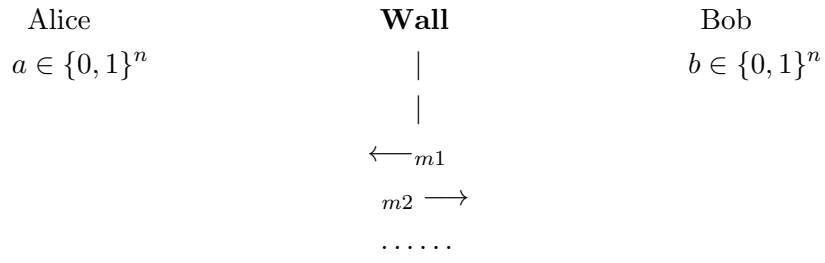
$$\mathrm{P}(x) = (x - \alpha_1)(x - \alpha_2)(x^3 + \ldots)(x^2 + \ldots)$$

where the terms on the right are irreducible. Note that here α_1, α_2 are roots. It is a fact that an irreducible polynomial has no roots. This can easily be seen by considering an irreducible polynomial $\mathrm{A}(x)$ and a root β. We can write $\mathrm{A}(x) = (x - \beta)\,\mathrm{Q}(x) + \mathrm{R}$. β is a root if and only if $R \equiv 0$, which would imply that $(x - \beta) |\, \mathrm{Q}(x)$. $\square$

We've known this since high school. Why is this important? The next section describes a concrete application in communication complexity.

9 Communication complexity

The basic setup of the communication complexity problem is as follows:



In the above diagram, Alice and Bob both have a bit string known only to themselves at the start. There is a wall between them, and they communicate by sending messages across the wall. Computation is free in this model, and we are only concerned with the number of bits communicated over the wall. Specifically, we want to minimize the number of bits sent. Note in general, we only ever need to send $n + 1$ bits, since Bob can send over his entire number, Alice can perform some computation on it, and Alice can send over 1 bit indicating yes/no depending on the problem at hand.

An example problem is equality, which is the question: Is $a = b$?

Any deterministic algorithm requires at least n bits of communication. Therefore, we look at randomized approaches to see if we can do better. We want a randomized strategy that outputs the correct answer with probability $1 - \frac{1}{n}$. This should work for every input pair a, b, and not just for randomized input. For example, a flaw with sending over random indexes and their values is that if a, b differ in only 1 bit, we would need to send a linear number of bits.

It is possible to do this with $O(\log n)$ bits of communication, using a one-sided error randomized algorithm. It's also known that $\Omega(\log n)$ bits is required, which implies that this is an asymptotically optimal solution.

Proof. We give an algorithm to do so here.

1) Alice fixes $\mathbb{F}_q$, where $q \in [n^2, 2n^2]$. She tells Bob, which sends $O(\log q) = O(\log n)$ bits.

2) Alice then forms $\mathrm{P}_A(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x$.

3) Bob also forms $\mathrm{P}_B(x) = b_n x^n + b_{n-1}x^{n-1} + \ldots + b_1 x$.

4) Alice choose $\alpha \in \mathbb{F}_q$ at random, sends α and $\mathrm{P}_A(\alpha)$ to Bob, which sends $O(\log n)$ bits.

5) Bob checks $\mathrm{P}_B(\alpha)$, and says $a = b$ if $\mathrm{P}_B(\alpha) = \mathrm{P}_A(\alpha)$.

There are two cases to analyze. If $a = b$, then $\mathrm{P}_A(x) - \mathrm{P}_B(x) = 0$, so $\mathbf{Pr}[\text{equal}] = 1$. If $a \neq b$, then the output is wrong if and only if α is a root of $\mathrm{P}_A(x) - \mathrm{P}_B(x)$. Since this polynomial has degree at most n, and we are choosing α from a field of size at least n^2,

$$\begin{aligned}\mathbf{Pr}[\text{output equal when not equal}] &= \mathbf{Pr}[\alpha \text{ is a root}] \\ &\leq \frac{n}{n^2} \\ &= \frac{1}{n}\end{aligned}$$

□

Note we could repeat this procedure to give us a probability of error at most $\frac{1}{n^k}$ (we could also just choose $q \in [n^{2k}, 2n^{2k}]$). This still maintains the $O(\log n)$ bound.

10 Polynomials as functions

Before we move on, let's make a brief note on interpreting polynomials as functions.

Fact: Every $\mathrm{P} \in \mathbb{F}[x_1, \ldots, x_n]$ computes some function $f : \mathbb{F}_q^n \to \mathbb{F}_q$.

Additionally, every function is computed by some polynomial (we can use interpolation to construct it). There exist q^{q^n} functions, but there are infinitely many polynomials. Therefore, there exist two polynomials that compute the same function.

If these are A and B, and if we let $\mathrm{Z} = \mathrm{A} - \mathrm{B}$,

$$\mathrm{Z} \text{ computes the 0 function}$$

The above is important because Z is a "non-zero" polynomial.

Example: $x^q - x \in \mathbb{F}_q[x]$ is always 0 since $\alpha^q = \alpha \pmod q$, for all $\alpha \in \mathbb{F}_q$.

One thing we might realize is that we can reduce any polynomial so that each individual degree is $\leq q-1$ *without* changing the computed function. For example, if $q = 2$, then $x^3 \equiv x^2 x \equiv x \pmod q$.

Question: How many reduced monomials are there?

Answer: Since there are n different x_i's, and each one can have a power p with $0 \leq p \leq q-1$, there are q^n reduced monomials.

Question: How many reduced polynomials are there?

Answer: Every reduced monomial can have a coefficient c with $0 \leq c \leq q-1$, so there are q^{q^n} reduced polynomials. This leads to the following corollary.

Corollary 10.1. *Every function $f : \mathbb{F}_q^n \to \mathbb{F}_q$ is computed by a unique reduced polynomial.*

Lastly, of particular interest might be the boolean functions $f : \{0,1\}^n \to \{0,1\}$.

11 Schwartz–Zippel Lemma

We now cover one last important fact about polynomials, based on the Degree Mantra. The Schwartz–Zippel Lemma was discovered independently by Schwartz and Zippel, as well as by others such as Lipton and Orr. We state a general form of it below (more general than the ones commonly found).

Schwartz–Zippel Lemma [Sch80, Zip79]:
Let $\mathrm{P} \in \mathbb{F}_q[x_1, \ldots, x_n]$ be a reduced polynomial (that is *not* the 0 function) with total degree $\leq d$. Then:

$$\mathop{\mathbf{Pr}}_{\alpha_1,\ldots,\alpha_n \sim \mathbb{F}_q}[\mathrm{P}(\alpha_1, \ldots, \alpha_n) \neq 0] \geq \frac{1}{q^{\lfloor \frac{d}{q-1} \rfloor}} \cdot \left(1 - \frac{d \bmod (q-1)}{q}\right)$$

The above might look scary (and it is!), but fortunately, we can reduce it in two common cases.

Case 1: $q = 2$
The right hand side simplifies to be 2^{-d}.

Case 2: $d < q - 1$
This corresponds to there being lots of choices for the α's. In other words, we have a big field to choose from. The right hand side simplifies to $1 - \frac{d}{q}$ in this case, which means that the probability that the polynomial evaluates to 0 is $\leq \frac{d}{q}$.

In fact, if we choose the α's from a subset $S \subset \mathbb{F}$, we have the more commonly stated version:

$$\mathbf{Pr}[\mathrm{P}(\alpha_1, \ldots, \alpha_n) = 0] \leq \frac{d}{|S|}$$

The Schwartz–Zippel Lemma can be proved by induction. For the base case $n = 1$, this is just the Degree Mantra. The induction is left as an exercise.

A small note: one should be careful in applying the Schwartz–Zippel Lemma. It only holds if the random point is chosen from a sufficiently large set. For example, consider the following polynomial in $\mathbb{F}_2[x_1, \ldots, x_n]$

$$x_1 \cdot x_2 \cdot \ldots \cdot x_n = \begin{cases} 1 & \text{if } x_i = 1,\ \forall x_i \\ 0 & \text{otherwise} \end{cases}$$

This non-zero polynomial is *most likely* zero for a randomly chosen point. However, it does not contradict the Schwartz–Zippel Lemma. Therefore it is important to consider the size of the field in relation to d before applying the lemma.

12 Perfect matching in bipartite graphs

We now present an application of the Schwartz–Zippel Lemma. Given a graph G, a perfect matching is a subset S of the edges such that every vertex appears in S exactly once. Furthermore, we have the restriction that G is bipartite.

In the 60's, Karp showed that this problem could be solved in $O(mn)$ time, where $m = |E|$ and $n = |V|$. This includes both the decision problem (does a perfect matching exist?) and the function problem (the problem of finding the perfect matching).

An open problem is the following: Can this be done in polylog(n) time "in parallel"? That is, is it in NC? Recall that NC is the set of problems that can be solved by circuits that are poly(n) in size with polylog(n) depth. (It is also not known if GCD can be done this way either).

If randomization is allowed, this problem is known to be in RNC. [Lov79]

Let U, V be the two sets of nodes in a bipartite graph. Clearly, $|U| = |V|$, otherwise there could not be a perfect matching. We can construct a square matrix with the nodes of U labeling the rows, and the nodes of V labeling the columns.

Let $e_{i,j}$ be the entry in the i^{th} row and j^{th} column of $U \times V$ matrix M.

$$\begin{pmatrix} & e_{i,j} & \\ & & \end{pmatrix}$$

Define the entry $e_{i,j} = \begin{cases} X_{ij} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$.
Here, we've emphasized with the capital X that X_{ij} is a variable, not a value.

Recall that the determinant of a matrix A is defined as:

$$\det(A) = \sum_{\pi \sim S_n} \text{sign}(\pi) \cdot \prod_{i=1}^{n} A_{i,\pi(i)}$$

Then the determinant of our matrix M can be interpreted as a polynomial. That is, $\det(M)$ is a polynomial in $\mathbb{Q}[X_{ij} : i \in u, j \in v]$ For our matrix M, we have:

$$\det(M) \begin{cases} = 0 & \text{if there is no perfect matching} \\ \neq 0 & \text{if there is a perfect matching} \end{cases}$$

To see this, note that the determinant chooses exactly one entry from every row and column. These can be interpreted as the matchings for those nodes (if the entry e_{ij} is chosen, then (i, j) is included in the perfect matching). Since the values are multiplied and assuming there is no cancellation (we don't show this here, but there is none), then the determinant is non-zero if and only if there is at least one permutation for which all the edges exist.

There's one caveat to this—we can't compute this determinant efficiently (to see if it is identically 0), since there are an exponential number of terms. Instead, we can use the Schwartz–Zippel Lemma to evaluate the determinant at a random point, choosing each $X_{ij} \in \{1, \ldots, n^2\}$. By computing the numeric determinant on a random point, we have:

$$\nexists \text{ a perfect matching} \implies \det(M) \text{ is always } 0$$

$$\exists \text{ a perfect matching} \implies \mathbf{Pr}[\det(M) \text{ evaluated on random point} = 0] \leq \frac{n}{n^2} = \frac{1}{n}$$

Lastly, the problem of computing the determinant over $\mathbb{Q}$ is in NC. [Csa76]

Additional note by scribe: The above matrix is called a Karp matrix. It can be generalized to non-bipartite graphs by constructing a matrix with $|V|$ rows and $|V|$ columns, where the entries are defined as:

$$e_{i,j} = \begin{cases} X_{ij} & \text{if } (i,j) \in E, \text{ and } j > i \\ -X_{ji} & \text{if } (i,j) \in E, \text{ and } i < j \\ 0 & \text{otherwise} \end{cases}$$

The relationship between the determinant of this matrix and the existence of a perfect matching is exactly the same as the Karp matrix (this is called Tutte's matrix).

13 Recommended reading

For further reading, the following are recommended:

http://people.csail.mit.edu/madhu/ST12/scribe/lect03.pdf

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-451-principles-of-digital-communication-ii-spring-2005/lecture-notes/chap7.pdf

http://shoup.net/ntb/ntb-v2.pdf

References

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of mathematics*, pages 781–793, 2004.

[Csa76] Laszlo Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, volume 79, pages 565–574, 1979.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

[Sho90] Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54(189):435–447, 1990.

[Zip79] Richard Zippel. *Probabilistic algorithms for sparse polynomials*. Springer, 1979.