8 Section Week 8 - Polynomials in Computing

This lecture doesn't assume any prior knowlege.

8.1 Fields

A field is a set $\mathbb{F}$ with two binary operations, addition and multiplication. Furthermore, the operations satisfy the field axioms:

- Both addition and multiplication are associative and commutative.
- There exists an additive identity 0 where for all $a \in \mathbb{F}$, $a + 0 = a$ and a multiplicative identity 1 where for all $a \in \mathbb{F}$, $a1 = a$.
- Every element $a \in \mathbb{F}$ has an additive inverse $-a$ such that $a + (-a) = 0$.
- Every nonzero element $a \in \mathbb{F}$ has a multiplicative inverse a^{-1} such that $aa^{-1} = 1$.
- Multiplication distributes over addition.

What are some examples of fields?

Is $\mathbb{R}$ a field? Is $\mathbb{Q}$ a field? Is $\mathbb{Z}$ a field? Is $\mathbb{C}$ a field? answers are yes yes no yes.

There's one more field that isn't often talked about in "non discrete math," but is very important for theory. It turns out that the integers modulo a prime p, written $\mathbb{F}_p$, are also a field and this tends to be important. The basic properties of multiplication and addition are the same as with the reals. The additive identity is $0 \mod p$ and the multiplicative identity is still $1 \mod p$. For any a, the additive inverse of a is $-a \mod p \equiv p - a \mod p$. For any a, the multiplicative inverse of a is b such that $ab \equiv 1 \mod p$. We use Fermat's little theorem to get that $a^{p-1} \equiv 1 \mod p$, so if $b = a^{p-2}$ then we get $ab = a^{p-1} \equiv 1 \mod p$.

If a problem calls for a prime number that is n binary digits long, how do we find one? Easy - pick a random n digit number and check if it's prime.

The prime number theorem states that the number of primes at most 2^n is about $\frac{2^n}{n}$, so if we pick $O(n)$ random n digit numbers and test them, with high probability we will find a prime number. Some examples of good primality tests are the Fermat primality test and Miller-Rabin primality test.

Thus, if we say "we do these operations in a large prime field $\mathbb{F}_p$," this is how such a field is instantiated.

8.2 Polynomials over Fields

Normal polynomials that we learn about in high school math are polynomials over the field of real numbers. However, we often care about polynomials over different kinds of fields so we must be rigorous in defining what a polynomial is.

We'll start with univariate polynomials. The set of univariate polynomials over a field $\mathbb{F}$ is written as $\mathbb{F}[x]$, and a degree d polynomial is of the form

$$p(x) = a_d x^d + a_{d-1} x^{d-1} \dots a_1 x + a_0$$

where $a_1, \dots a_d \in \mathbb{F}$.

If we want to evaluate a polynomial over a field, we do the exact same thing as with the real numbers. We simply plug in x in each location and evaluate the whole thing. Multiplication, addition, and exponents are done as their field operations.

Also useful is that the fundamental theorem of algebra applies has an analog for finite fields as well! A degree d polynomial in $\mathbb{F}[x]$ has at most d roots.

Multivariate polynomials are a little more complicated. The set of multivariate polynomials over a field F with indeterminate variables $x_1, \dots x_n$ is $\mathbb{F}[x_1, \dots x_n]$. A multivariate polynomial over an infinite field can have in-

finitely many roots (e.g. $x^2 - y^2$), and for finite fields we have something called the Schwartz Zippel lemma which we will not state or use here.

Application of polynomials to Communication Complexity: let's compute $f(a, b) = 1$ if $a = b$. Recall that with private randomness we showed that it can be done with $O(\log n)$ bits of communication.

Now Alice and Bob do the following: Alice picks a prime $q > n^2$ and sends it to bob with $O(\log n)$ bits. Alice and Bob now think about the polynomials $P_A(x) = a_0x^0 + \cdots + a_{n-1}x^{n-1}$ and $P_B(x) = b_0x^0 + \cdots + b_{n-1}x^{n-1}$. Note that $P_A - P_B$ is the zero polynomial if and only if $a = b$, so Alice and Bob want to check if $(P_A - P_B)(x) = 0$ for all x.

Instead, Alice randomly chooses an $\alpha \in \mathbb{F}_q$ and sends $(\alpha, P_A(\alpha))$ to Bob with $O(\log n)$ bits. Bob confirms if $P_B(\alpha) = P_A(\alpha)$.

Since $P_A - P_B$ is univariate degree n, it has at most n roots. Thus, if $P_A - P_B = 0$ then this protocol always correctly finds that $a = b$ and outputs $f(a, b) = 1$. If $P_A - P_B \neq 0$, then $P_A - P_B$ is a polynomial that evaluates to zero on at most n out of n^2 field values, so if $a \neq b$ the probability this protocol fails is the probability we unluckily pick one of the roots, which is $\frac{1}{n}$.

8.3 Polynomials as Functions

When you really think about it, every polynomial in $\mathbb{F}[x_1, \ldots x_n]$ just computes a function $f : \mathbb{F}_q^n \to \mathbb{F}_q$. There are infinitely many polynomials (since there's currently nothing bounding the degree). So there must necessarily be functions that are computed by more than one polynomial. Can every function be computed by a polynomial?

The answer is yes, by Lagrange interpolation.

Suppose we write a point as $z^1 = (x_1^1, x_2^1, \ldots x_n^1, y^1)$ and we wish to find a polynomial that passes through some m distinct points $z^1, z^2, \ldots z^m$. We write $(x_1^i, \ldots x_n^i) = \overline{x^i}$. This polynomial is:

$$L(\overline{x}) = \sum_{i=1}^{m} y^i \ell^i(\overline{x})$$

where intuitively $\ell^i(\overline{x^i}) = 1$ but $\ell^j(\overline{x^i}) = 0$ for all $j \neq i$. Thus, when we plug in we get $L(\overline{x^i}) = y^i$.

The form of ℓ^i is:

$$\ell^i(\overline{x}) = \frac{(\overline{x} - \overline{x^1})}{(\overline{x^i} - \overline{x^1})} \cdots \frac{(\overline{x} - \overline{x^{i-1}})}{(\overline{x^i} - \overline{x^{i-1}})} \frac{(\overline{x} - \overline{x^{i+1}})}{(\overline{x^i} - \overline{x^{i+1}})} \cdots \frac{(\overline{x} - \overline{x^m})}{(\overline{x^i} - \overline{x^m})}$$

Interpolation is an important idea.

Thus, given any $m = q^n$ points specifying the output on every input to a function f, we get a polynomial that computes f.

8.4 Arithmetization of Logic

This is all well and good but as you can imagine such a large polynomial is pretty gross to deal with, and we may ask natural questions like what do polynomials as a computational model look like. Well it turns out that there's a pretty natural way to turn any boolean formula (not circuit) into a polynomial.

Recall that a boolean formula uses AND, OR, NOT gates with out degree 1. I claim that given any boolean formula with n inputs, I can construct a polynomial $p \in \mathbb{F}_2[x_1, \ldots x_n]$ that computes the same function.

We do this by induction. Our base case is that the input x_i just corresponds to the polynomial $p(\overline{x}) = x_i$.

Suppose we have some AND gate, and the two inputs to AND have already been expressed as polynomials $p(x)$ and $q(x)$. I claim that $p(x)q(x) = p(x)ANDq(x)$. We can quickly verify this with a truth table.

Similarly, the OR gate is equivalent to $1-(1-p(x))(1-q(x))$ and the NOT gate is equivalent to $1-p(x)$. Thus, any boolean formula can be written as a polynomial

What is the degree of our polynomial? The output of a NOT gate has the same degree as the input, so we only care about the AND and OR gates. When we multiply two (even multivariate) polynomials we simply add their degrees together, so by induction the degree of our final output is just the number of times a variable of the input is used. If all inputs are used exactly once, our polynomial which computes the formula has degree n.

This has a close connection to satisfiability as well - suppose we have some 3SAT instance like $\phi(x) = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_4 \vee x_5) \ldots$. If we have m clauses and turn this into a degree $3m$ polynomial $P_\phi(x) = \phi(x)$, then we get that ϕ is satisfiable if there exists an input x such that $P_\phi(x) = 1$. This may lead to the idea that we can just check if P_ϕ is a zero polynomial, but unfortunately this is not so easy. For example, $x^2 + x$ is a nonzero polynomial that computes the zero function.

We can also count the number of satisfying assignments to ϕ by computing $\sum_{x \in \{0,1\}^n} P_\phi(x)$. This is called the $\#SAT$ problem - given a boolean formula (for example, a 3-CNF), how many satisfying assignments are there? We call these kinds of problems counting problems. Furthermore, $\#SAT$ is complete for the class $\#P$, which is counting the number of accepting paths for a nondeterministic poly time TM.

Expressing $\#SAT$ as a polynomial is an essential component in a proof called $IP = PSPACE$, which is very important but the full details are outside the scope of this lecture.

8.5 Fourier Analysis of Boolean Functions

We will now briefly mention that Fourier analysis of boolean functions is an important topic, and will maybe give a more in depth treatment of it in a later lecture.

The idea of Fourier analysis is to interpolate a boolean function with a polynomial over the real numbers. We view a function as $f : \{-1, +1\}^n \to \mathbb{R}$. This allows us to interpolate (say, a function over 3 variables) as

$$\left(\frac{1}{2} - \frac{1}{2}x_1\right)\left(\frac{1}{2} - \frac{1}{2}x_2\right)\left(\frac{1}{2} - \frac{1}{2}x_3\right) \cdot f(-1,-1,-1) +$$
$$\vdots$$
$$\left(\frac{1}{2} + \frac{1}{2}x_1\right)\left(\frac{1}{2} + \frac{1}{2}x_2\right)\left(\frac{1}{2} + \frac{1}{2}x_3\right) \cdot f(+1,+1,+1)$$

Once we take a polynomial like this and FOIL it out, we get a multivariate polynomial, which is also multilinear because the exponent for each variable is at most 1, of the form:

Theorem (Multilinear Representation of Boolean Functions)**.** *Every boolean function $f : \{\pm 1\}^n \to \mathbb{R}$ is uniquely expressible as a multilinear polynomial of the form*

$$\sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i$$

where $\hat{f}(S)$ is a coefficient determined by the function f. This is called the "Fourier Expansion" of f, and $\hat{f}(S)$ is called the Fourier coefficient of S. Notice that the product is just the parity of set S, so this is an expression of f as a linear combination of parity functions over subsets of $[n]$.