Lecture 17: Finite fields, Random self-reducibility of permanent, IP

Anup Rao

May 21, 2024

Finite fields

An important fact that we will use a few times is that the integers modulo a prime number p are a *finite field*. This means, that not only can you add and multiply elements modulo p, you can divide:

Fact 1. *Given a prime p, for every $a \neq 0 \mod p$, there is a number b such that*

$$ab = 1 \mod p.$$

Proof By Euclid's greatest common divisor algorithm, the greatest common divisior of x, y can be expressed as $bx + cy$, for two integers b, c. Apply that result to a, p. Since the greatest common divisor of a, p is 1, we get

$$1 = ba + cp,$$

which proves that $ab = 1 \mod p$. ■

This ability to do division means that a lot of the results that we prove over the real numbers also apply over the finite field $\mathbb{F}_p$ (the integers modulo p). For example, every degree d univariate polynomial over $\mathbb{F}_p$ has at most d roots, and the Schwartz-Zippel lemma holds as well.

Polynomial interpolation

We have talked about how every univariate degree d polynomial has at most d roots. This fact can be generalized to the following statement:

Fact 2. *Given $d+1$ distinct elements $a_1, \ldots, a_{d+1} \in \mathbb{F}$ and $b_1, \ldots, b_{d+1} \in F$, there is a unique degree d polynomial f such that $f(a_i) = b_i$ for all i.*

Proof First, we construct a degree d polynomial f with the required properties:

$$f(x) = \sum_{i=1}^{d+1} b_i \cdot \frac{\prod_{j \neq i}(x - a_j)}{\prod_{j \neq i}(a_i - a_j)}.$$

This is a degree d polynomial satisfying all the constraints. Now, to show that it is unique, suppose there was another degree d polynomial g that had the same properties. Then $g - f$ is a degree d

polynomial with $d+1$ roots, since for every i, $g - f$ evaluates to 0 on input a_i. That contradicts the fact that a non-zero degree d polynomial can have at most d roots. ■

The Permanent is randomly self-reducible

Recall that the permanent of an $n \times n$ matrix M is defined to be

$$\mathsf{perm}(M) = \sum_{\pi} \prod_{i=1}^{n} M_{i,\pi(i)},$$

where the sum is taken over all permutations $\pi : [n] \to [n]$.

Recall that the permanent is important because it is a complete function for the class **#P** (discussed in the last lecture).

Here is an interesting observation: if you can compute the permanent with high probability on a random matrix with entries from $\mathbb{F}_p$, then you can also compute it with good probability on an *arbitrary* matrix with entries from $\mathbb{F}_p$. Indeed, given a $n \times n$ matrix M, let R be a matrix with random entries from $\mathbb{F}_p$. Then the function

$$f(x) = \mathsf{perm}(M + x \cdot R)$$

is a univariate polynomial of degree n. But each of the values

$$f(1), f(2), \ldots, f(n+1)$$

is the evaluation of the permanent on a uniformly random matrix. So, if we have an algorithm that computes the permanent on random matrices with probability of error $1/(3(n+1))$, this algorithm will compute all of the values of

$$f(1), \ldots, f(n+1)$$

with probility of error at most $1/3$. From these values, we can reconstruct the f using polynomial interpolation, and then evaluate $f(0) = \mathsf{perm}(M)$, to obtain the desired permanent.

Interactive proofs

One way to define **NP** is via the idea of a proof system. **NP** is the set of functions f for which there is a polynomial time verifier algorithm V such that given any x with $f(x) = 1$, there exists a prover P that can prove to the verifier that $f(x) = 1$ by providing a polynomial sized witness w for which $V(x, w) = 1$, yet if $f(x) = 0$, no such prover exists.

What happens if we allow the verifier to have a longer *interactive* conversation? Presumably, giving the verifier the ability to adaptively ask the prover questions based on his previous responses should give the verifier more power, and so allow the verifier to verify the correctness of the value for a larger set of functions. In fact, this does *not* give the verifier additional power: for if there is such an interactive verifier V^I for verifying that $f(x) = 1$, we can design a non-interactive verifier that does the same job. The new verifier will demand that the prover provide the entire transcript of interactions between V^I and a convincing prover. The new verifier can then verify that the transcript is correct, and would have convinced V^I. Thus, if f has an interactive verifier, then $f \in \mathbf{NP}$.

The story is more interesting if we allow the verifier to be randomized. We say that $f \in \mathbf{IP}$ if there is a polynomial time randomized verifier V such that

Completeness For all x, if $f(x) = 1$, there is an oracle P such that $\Pr_r[V^P(x,r) = 1] \geq 2/3$.

Soundness For all x, if $f(x) = 0$, for every oracle P, $\Pr_r[V^P(x,r) = 1] \leq 1/3$.

Since any prover can be simulated in polynomial space, if $f \in \mathbf{IP}$, then $f \in \mathbf{PSPACE}$. The algorithm for f can just try all possible sequences of messages from the prover until it finds a sequence of messages that convinces the verifier, if such a sequence exists.

Theorem 3. $\mathbf{IP} \subseteq \mathbf{PSPACE}$.

It is easy to check that allowing the prover to be randomized does not change the model.

We shall eventually prove that $\mathbf{IP} = \mathbf{PSPACE}$ (and so **IP** is potentially much more powerful than **NP**).

Example: Graph non-Isomorphism

Two graphs on n vertices are said to be *isomorphic* if the vertices of one of the graphs can be permuted to make the two equal.

Consider the problem of testing whether two graphs are *not* isomorphic: the boolean function f such that $f(G_1, G_2)$ is 1 if and only if G_1 is not isomorphic to G_2. $f \in co\mathbf{NP}$, since the prover can just send the verifier the permutation that proves that they are isomorphic. We do not know if $f \in \mathbf{NP}$, but it is easy to prove that $f \in \mathbf{IP}$.

Here is the simple interactive protocol:

1. The verifier picks a random $i \in \{1,2\}$.

2. The verifier randomly permutes the vertices of G_i and sends the resulting graph to the prover.
3. The prover responds with $b \in \{1, 2\}$.
4. The verifier accepts if $i = b$.

If G_1, G_2 are not isomorphic, then any permutation of G_i determines i, so the prover can determine i and send it back. However, if G_1, G_2 are isomorphic, then the graph that the prover receives has the same distribution whether $i = 1$ or $i = 2$, thus the prover can guess the value of i with probability at most $1/2$. Repeating the protocol several times, the verifier can make the probability of being duped by a lying prover exponentially small.