6.S897 Algebra and Computation Feb 4, 2015

Lecture 1

Lecturer: Madhu Sudan *Scribe: Itay Berman*

1 Administrative Topics

- Website: `http://people.csail.mit.edu/madhu/ST15/`.
- Mailling list – if you are not in it, please email Madhu with your details.

1.1 Responsibilities

1. Scribe: each student in the course (registered, listener or attending just for fun) must scribe at least one lecture. Please sign up for dates to scribe (see instructions in the course website).
2. Project: will mainly include reading and presenting a new topic to the class. A written report is also required, but no new results are expected.
3. Participation: please participate in class.
4. Problem Sets: maybe will be, and if so then maybe be submitted.

2 Motivating Example — Factoring Polynomials [1]

Consider the following question: Given $p_1(x), p_2(x) \in \mathbb{F}[x]$ (where $\mathbb{F}[x]$ is the polynomial ring in formal variable x over the field $\mathbb{F}$, i.e., the set of polynomials in x whose coefficients are elements of $\mathbb{F}$) with $deg(p_1), deg(p_2) < n$ and the sets $\{p_1(\alpha_1), p_2(\alpha_1)\}, \{p_1(\alpha_2), p_2(\alpha_2)\}, \ldots, \{p_1(\alpha_{10n}), p_2(\alpha_{10n})\}$, where $\{\alpha_i\}_{i=1}^{10n}$ are distinct elements in $\mathbb{F}$, can you compute p_1 and p_2?

If we could have known in the i'th set which element corespondents to evaluation of which polynomial on α_i, we can simply interpolate and get both p_1 and p_2. We can, however, compute $A = p_1 + p_2$ and $B = p_1 \cdot p_2$ without knowing this correspondence. We cen now define the bivariate polynomial $q \in \mathbb{F}[x, y]$ as

$$q(x, y) \overset{\text{def}}{=} (y - p_1(x)) \cdot (y - p_2(x)) = y^2 - A(x) \cdot y + B(x).$$

Note that the roots of q with respect to y are exactly p_1 and p_1, so by the quadratic formula we can write

$$y = \frac{A \pm \sqrt{A^2 - 4B}}{2}.$$

But can we really use the quadratic formula for bivariate polynomials as we did? Can we divide by 2 in the field? What does square root means? It turns out that we can answer the above questions, and to show that this approach works.

What if we have more polynomials, say $p_1, p_2, \ldots, p_{10}$? We can define again

$$q(x, y) \overset{\text{def}}{=} (y - p_1(x)) \cdot (y - p_2(x)) \cdot \ldots \cdot (y - p_{10}(x)) = y^{10} - A_1(x) \cdot y^9 + A_2(x) \cdot y^8 + \cdots + A_{10}(x).$$

Now, q is no longer quadratic in y, and we don't have a formula to use to factor it. Can we solve this problem? Can we do so efficiently? The above questions and more will be addressed in this course.

3 Course Topics

3.1 Primer on Algebra

Some examples:

1. degree n polynomials has at most n roots (this theorem is used everywhere in CS).
2. $(x+y)^p = x^p + y^p$ over fields with characters p.
3. spare polynomial with many roots: $\prod_{\alpha \in \mathbb{F}_q}(x-\alpha) = x^q - x$, where $|\mathbb{F}_q| = q$.

3.2 Algorithms for Polynomials

1. addition, multiplication, division, GCD.
2. Factorization:
   (a) univariate polynomials factorization over finite fields.
   (b) univariate polynomials factorization over rationals.
   (c) multivariate polynomials factorization.
   (d) (root finding over real polynomials.)

   We'll see that the relation between the above algorithms relies on a connection between Z to $\mathbb{F}[x]$.
3. primality testing (much more algebra than number theory).
4. solving systems of polynomial equations (NP-hard): Grobner basis, ideal membership.

3.3 Complexity

1. Models: circuits and formulas.
2. Determinant (easy) vs. Permanent (hard).
3. $P = NC$ (in the algebraic sense), or small circuits are equivalent to small depth.
4. Lower bounds.
5. Polynomial identity testing: we know how to do it with randomized algorithms, but not deterministic ones (how to prove a polynomial is the zero polynomial?)

4 Membership in Permutation Group

You are given a cube and a set of legal moves. Can you transform the cube, using only legal moves, to some specific configuration? This question is exactly the problem of membership in permutation group.

Let $[n] \overset{\text{def}}{=} \{1, \ldots, n\}$, let S_n denote the set of permutations from $[n]$ to $[n]$, let $e \in S_n$ denote the identity permutation, and for $f, g \in S_n$ let $(g \cdot f)(i) \overset{\text{def}}{=} g(f(i))$. For $T \subseteq S_n$, let $\langle T \rangle$ denote the group generated by T with respect to operation '$\cdot$'.

Definition 1 (Permutation Group Membership) *Given $T \subseteq S_n$ such that $T = \{t_1, t_2, \ldots, t_k\}$ and $\sigma \in S_n$, decide if $\sigma \in \langle T \rangle$ (in time* poly*(n,k)).*

We we'll see how to implement the following natural strategy. Our goal is to transfer the identity permutation e (which always belongs to $\langle T\rangle$) to σ using only moves from T. Namely, we want to find $T_1, T_2, \ldots, T_r \in T$ such that $e \cdot T_1 \cdot \ldots \cdot T_r \equiv \sigma$ with the following restrictions. Each T_i fixes the first $i-1$ elements of $[n]$ (i.e., $T_i(j) = j$ for $j < i$) and transfer $\sigma^{-1}(i)$ to i (i.e., $T_i(\sigma^{-1}(i)) = i$). What guarantees that such T_i's exists? If T_1 does not exists, then $\sigma \notin \langle T\rangle$ (since σ itself meets T_1's requirements). If $T_1, \ldots, T_i$ exist, then if T_{i+1} does not exists, it follows that $\sigma \notin \langle T\rangle$ (since $\sigma \cdot T_1^{-1} \cdot \ldots \cdot T_i^{-1}$ meets T_{i+1}'s requirements).

So, if we can efficiently find these T_i's, or determined that there are none, we can solve the permutation group membership problem. To do so we use the following generators.

Definition 2 *A permutation $\pi \in S_n$ has type$(\pi) = i$ if $\pi(j) = j$ for every $j \in [i-1]$ and $\pi(i) \neq i$.*

Definition 3 *A set $S \subseteq \langle T\rangle$ is* Strong Generating Set for $\langle T\rangle$ *if*

$$\forall i < j\colon (\exists \pi \in \langle T\rangle \ \ s.t. \ type(\pi) = i \wedge \pi(j) = i) \implies \exists! \tau \in S \ s.t. \ type(\tau) = i \wedge \tau(j) = i$$

Note that the size of every strong generating set is at most $\binom{n}{2}$.

Given a strong generating set for $\langle T\rangle$, we can use the above approach to find these T_i's, or to determined that $\sigma \notin \langle T\rangle$. It is left to show how to find such strong generating set. For $S \subset S_n$ let $|S\rangle = \{\pi_r \cdot \pi_{r-1} \cdot \ldots \cdot \pi_1 \colon \pi_i \in S \wedge type(\pi_i) > type(\pi_{i-1})\}$ and for $\pi \notin |S\rangle$ let $FinalEle(\pi, S)$ denote the final permutation generated by taking the previous permutation, starting with π, and compose it with a permutation from S such that the resulting permutation has larger type than the previous permutation.

```
**Sims' Algorithm** $(T)$
$S \leftarrow \emptyset$
**while** $\exists \sigma, \tau \in S \cup T$ s.t. $\sigma \cdot T \notin |S\rangle$:
    $S \leftarrow FinalEle(\sigma \cdot \tau, S) \cup S$.
```

At the end of Sims' algorithm, the set S is strong generating set for $\langle T\rangle$ (we did not show this in class).

References

[1] Sigal Ar, Richard J. Lipton, Ronitt Rubinfeld, and Madhu Sudan. Reconstructing algebraic functions from mixed data. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 503–512, 1992.