6.S897 Algebra and Computation February 13, 2012

Lecture 2

Lecturer: Madhu Sudan *Scribe: Zeyuan Allen Zhu*

1 Administrivia

- Students signed them up on a piece of paper today. If you missed the first class, please subscribe to the course mailing list yourself.
- Please sign-up for scribing by sending an email to Prof. Sudan. All students (including listeners) need to scribe at least once.
- Lectures always start on time, so do not come late to the classroom.
- Will have one or two problem sets.
- Madhu will select topics beyond the course materials for students to present as course projects. That requires each student to spend 10 minutes to explain the work in class, and complete a write-up that can be useful for others. It is a casual and fun job, and encouraged to done in pairs.

2 Membership Problem in Permutation Groups

Instead of listing potential topics that we plan to cover in this course now, we start with a concrete example.

How can we solve a cube?

(In class Prof. Sudan handed to the students a fancy cube that few of us have seen before, found in Prof. Erik Demaine's office.) This is an algorithmic challenge to us. We have this cubic object, and can do a few moves on it. Can we reach some given state from the beginning, and how? We now formalize a broader class of such problems in a purely mathematical way.

Suppose that we are given finite number of n pieces and there are some basic moves allowed. In a traditional cube, there are $n = 54$ pieces (stickers) and we can take for instance the top or side face and rotate. We introduce now a few notions before we proceed:

- $[n] := \{1, 2, \dots, n\}$.
- $S_n := \{\pi : [n] \to [n] : \pi \text{ is one-to-one}\}$.
- $\langle S \rangle := \{\text{group of permutations obtained by composing members of } S\}$.

Suppose that the operations allowed are permutations $\pi_1, \pi_2, \dots, \pi_k \in S_n$, and we can do any combination of them. Let $S = \{\pi_1, \pi_2, \dots, \pi_k\}$ and $\langle S \rangle$ is a subgroup of S_n. Given an arbitrary permutation $\pi \in S_n$, does $\pi \in \langle S \rangle$? Can we solve this problem efficiently?

In general, we consider cases where $k = |S| = O(n)$ and we expect *efficient* algorithms to run in time $\mathsf{poly}(n)$.[1] We note that it may take a very large number of compositions to generate $\langle S \rangle$. For instance, it is an exercise to given an example $S \subseteq S_n$ such that $\exists \sigma \in \langle S \rangle$ while

$$\min_{l}\{l \text{ s.t. } \sigma = s_1 s_2 \cdots s_l \wedge \forall i, s_i \in S\} = \exp(n^{1/10}) \ .$$

In order to overcome this, we need the idea of an arithmetic circuit. We will not use much of this notion today, but the definition is as follows:

[1]Typically even writing done a permutation requires $O(n)$ time. There are other ways to represent permutations in a more concise way, and thus one may also expect the existence of algorithms that runs in time $\mathsf{polylog}(n)$, though a little too ambitious.

2.1 Arithmetic Circuit over Group

(In the future we will also consider it over rings or fields.)

- Input: $S \subseteq G$.
- Circuit: $C = (t_1, t_2, \ldots, t_m)$ a collection of group elements, such that for each t_i:
  - either $t_i \in S$,
  - or $t_i = t_j \cdot t_k$ such that $j, k < i$.
- We say that C computes T if $T \subseteq \{t_1, t_2, \ldots, t_m\}$.

In this language, our question can be formalized as:

- Input: n, $S \subseteq S_n$, and a permutation $\pi \in S_n$.
- Output: YES if there exists an arithmetic circuit computing $\{\pi\}$ using input S, or NO if there isn't any.

We remark here that such a question is general itself because:

Fact 1 *Every finite group is a (subgroup of) permutation group.*

Proof Given a finite group G with $|G| = n$, we consider the following multiplication map:

$$g \in G \text{ is mapped to } \pi_g \in S_n, \text{ where } \pi_g : G \to G \text{ by } \alpha \mapsto g\alpha.$$

It satisfies that $\pi_a \pi_b = \pi_{ab}$, and every element is mapped to a permutation whose length is as large as the group itself. ■

A big challenge in group theory is to ask that: is there a way to represent an element by not n numbers but perhaps $\log n$ or $\mathsf{polylog}(n)$ numbers? For instance, can we solve the membership question without going through this intermediate permutation group for certain problems?

2.2 The Sims Algorithm

(Most of the materials are following a book *Permutation Group Algorithms* by Ákos Seress.)

Let us define $\mathcal{C}_{k,j} = \{\pi | \pi(i) = i \forall i < k, \text{ and } \pi^{-1}(k) = j\} \subset S_n$, and notice that $\mathcal{C}_{k,j} \cap \mathcal{C}_{k',j'} = \emptyset$ as long as $(k, j) \neq (k', j')$. We always assume that $j > k$. Now, the entire space of permutation group S_n is partitioned by such $\mathcal{C}$'s:

$$e \cup \bigcup_{1 \le k < j \le n} \mathcal{C}_{k,j} = S_n \ .$$

We say that the k-value of a permutation π is k_0 if $\pi \in \mathcal{C}_{k_0,j}$ for some $j > k_0$. We further define (and notice that T can be any set, not necessarily a subgroup of S_n):

Definition 2

- Minimal set*: T is minimal if $|T \cap \mathcal{C}_{k,j}| \le 1$ for all k, j.*
- Representative set*: $T \subseteq G$ is a representative set for $G = \langle S \rangle$, if $|T \cap \mathcal{C}_{k,j}| \ge 1 \Leftrightarrow |G \cap \mathcal{C}_{k,j}| \ge 1$ for all k, j.*
- Strong generating set (SGS)*: T is an SGS for G if T is both minimal and representative for G.*

Fact 3 *For any group G, there exists SGS T, and $|T| \le \binom{n}{2}$.*

2.3 Membership Testing Given SGS

Given π and suppose we want to check if $\pi \in G = \langle S \rangle$. We have that $\pi \in \mathcal{C}_{k,j}$ is in some unique $\mathcal{C}_{k,j}$. If $T \cap \mathcal{C}_{k,j} = \emptyset$ we are done, because this implies $G \cap \mathcal{C}_{k,j} = \emptyset$ and thus it cannot be the case that $\pi \in \mathcal{C}_{k,j}$.

Otherwise, we can find some $\sigma \in T \cap \mathcal{C}_{k,j}$. So can apply $\sigma^{-1}\pi$ and verify if $\sigma^{-1}\pi \in G$. This verification suffices because: 1) $\sigma^{-1}\pi$ will have a strictly smaller k-value than π; 2) $\pi \in G$ if and only if $\sigma^{-1}\pi \in G$, as σ is an element in G.

2.4 Computing Strong Generating Set

Let us start with a simple trial of computing SGS.

- Start with $T \leftarrow \emptyset$.
- INSERT(σ, T)
  - If σ is identity, we stop.
  - If $\sigma \in T$, we stop.
  - If $\sigma \in \mathcal{C}_{k,j}$ and $T \cap \mathcal{C}_{k,j} = \emptyset$, $T \leftarrow T \cup \{\sigma\}$.
  - If $\sigma \in \mathcal{C}_{k,j}$ and $T \cap \mathcal{C}_{k,j} = \{\tau\}$, INSERT$(\tau^{-1}\sigma, T)$.

Definition 4 *Canonical product is the product of previously inserted elements whose k-values are increasing.*

It satisfies that if two canonical products of elements from minimal sets are identical, i.e., $\sigma_1\sigma_2 \cdots \sigma_l = \tau_1\tau_2 \cdots \tau_m$ while $\sigma_1, \ldots, \sigma_l, \tau_1, \ldots, \tau_m \in T$ and T is a minimal set, then $l = m$ and for all i, $\sigma_i = \tau_i$.

Definition 5 *Closure$(T) = \bar{T} = \{$canonical products of elements of $T\}$.*

In this language, what we have shown earlier regarding having an SGS we can test membership, is equivalent to:

$$T \text{ is an SGS} \Rightarrow \bar{T} = G.$$

Fact 6 *INSERT(σ, T) will make sure that $\sigma \in \bar{T}$ after the procedure.*

And this above fact suggests that we can insert all elements from S into T and this ensures that $S \subseteq \bar{T}$. This does not fully answer our question yet, since we still need to show that $\bar{T}$ is a subgroup of S_n (i.e., closed under multiplications). Ideally, we can fix this by taking two elements from the closure $\bar{T}$ and try to insert it, but unfortunately there will be too many such elements in $\bar{T}$. So, instead we consider the following algorithm:

Computing SGS (n, S)**:**

- Start with $T \leftarrow \emptyset$.
- While there exists $\sigma \in S$ or $\sigma \in T \cdot T$, such that INSERT(σ, T) is a non-trivial operation (i.e., it changes T), do it.

This algorithm terminates in polynomial time because T is a minimal set and has no more than $\binom{n}{2}$ permutations. Its correctness will be shown with the help of the next lemma:

Lemma 7 *If $T \cdot T \subseteq \bar{T}$, then $\bar{T} \cdot \bar{T} = \bar{T}$ forms a subgroup.*

It is obvious that if the above lemma holds (i.e., $\bar{T} \cdot \bar{T} = \bar{T}$ is a subgroup), and we know that $\hat{T}$ contains S, it must be that $\bar{T} = \langle S \rangle$. To show that T is an SGS, we only need to notice that for all k, j, such that $\langle S \rangle \cap \mathcal{C}_{k,j} \neq \emptyset \Rightarrow \bar{T} \cap \mathcal{C}_{k,j} \neq \emptyset \Rightarrow T \cap \mathcal{C}_{k,j} \neq \emptyset$, using the property of canonical products.

Therefore, the only thing left is to prove Lemma 7.

Proof [of Lemma 7] We prove by induction.[2]

Take two elements $\sigma, \tau \in \bar{T}$ and we want to show that $\sigma\tau \in \bar{T}$. Assume that $\sigma = \sigma_1 \cdots \sigma_l$ is the canonical representation with $\sigma_i \in T$ and $k(\sigma_i) < k(\sigma_{i+1})$. Here we use $k(\cdot)$ to represent the k-value of a permutation. Similarly we assume that $\tau = \tau_1 \cdots \tau_m$. We prove by double induction on the decreasing of $k = k(\tau_1)$ and then on the increasing of l.

If $k = n$ this means $\tau = e$ is the identity permutation, and the claim trivially holds.

If $k < n$ and $k(\sigma_l) < k = k(\tau_1)$, then $\sigma_1 \ldots \sigma_l \tau_1 \ldots \tau_m$ is of the increasing k-value and thus in $\bar{T}$.

If $k < n$ but $k(\sigma_l) \geq k = k(\tau_1)$, then since $T \cdot T \subseteq \bar{T}$, we can write $\sigma_l \tau_1 = \alpha_1 \cdots \alpha_p \in \bar{T}$ where α_i's are of increasing k-values.

$$\begin{aligned} \sigma\tau &= \sigma_1 \cdots \sigma_{l-1} \quad \underbrace{\sigma_l \tau_1}_{\overset{\parallel}{\alpha_1 \alpha_2 \cdots \alpha_p}} \quad \tau_2 \cdots \tau_m \\ &= \sigma_1 \cdots \sigma_{l-1} \quad \alpha_1 \alpha' \quad \tau' \end{aligned} \tag{1}$$

In fact, since $k(\sigma_l) \geq k$ and $k(\tau_1) = k$, we have that $k(\alpha_1) \geq k$ while all $k(\alpha_2), \ldots, k(\alpha_p) > k$. If we define $\tau' = \tau_2 \cdots \tau_m$ while $\alpha' = \alpha_2 \cdots \alpha_p$, we have that by induction $\alpha'\tau' \in \bar{T}$ since $k(\tau_2) > k$. Furthermore, since $k(\alpha'\tau') > k$ (as both $k(\alpha'), k(\tau') > k$), we know $\alpha_1 \alpha' \tau' = \sigma_l \tau \in \bar{T}$. At last, having that $k(\sigma_l \cdot \tau) \geq k$, and by induction on l, we have that $\sigma\tau = (\sigma_1 \cdots \sigma_{l-1})(\sigma_l \tau) \in \bar{T}$. We conclude that $\bar{T}$ is closed under multiplication. ■

2.5 Other Works

Babai and Codenotti (FOCS'08) have also been looking at other representations of groups. For instance, for subsets of matrices $\mathbb{F}_q^{n \times n}$. They achieved efficient algorithms except for two caveats: one needs to be able to factor certain numbers such as $q - 1$ or $q^5 - 1$, and do discrete log over $\mathbb{F}_q$.

3 Contents of the course

- Berlekamp's algorithm for factoring univariate polynomials over $\mathbb{F}_p$ for large prime p. This algorithm was introduced in 1972 and is randomized. It remains an open question to factor even simple polynomials like $x^2 - a$ in deterministic polynomial time (i.e., $\mathsf{polylog}(p)$).
- Finite fields.
- (Factoring) multivariate polynomials.
- Factorization over rational polynomials (LLL).
- Techniques to solve multivariate systems of polynomial equations.
- Two different models for algebraic computations.

[2] The scribe today wants to point out here that this proof was not covered in class and written with the help from the course notes in 2005. However, in 2005 the k-value was defined in an opposite way which leads to a different inductive proof.

- Blum-Shub-Smale model of computing. A hard problem under this model is given polynomials $P_1, \ldots, P_m \in \mathbb{C}[x_1, \ldots, x_n]$ with coefficients being 0-1's, and to check if they have a common zero.
- Valiant model of computing. A hard problem under this model is the permanent of matrices.

4 Assignment

Email madhu@mit.edu with your favorite and surprising algorithm and explain why. Due by the end of this week.