**Graph Isomorphism course, Spring 2017**
Instructor: László Babai
Notes by Angela Wu and instructor
Thursday, May 4, 2017

# 12 Day 12, ThWk6

Recall our discussion of StringIso from last class. We consider strings $x : \Omega \to \Sigma$, which are functions from $\Omega$ (the set of positions) to $\Sigma$ (the alphabet).

String Isomorphism (StringIso)
**Input:** Strings $x, y \in \Sigma^\Omega$ and a group $G \leq \mathrm{Sym}(\Omega)$. ($G$ is given by a list of generators.)
**Decision problem:** Does there exist $\sigma \in G$ such that $x^\sigma = y$.
**Computation problem:** Compute $\mathrm{ISO}_G(x, y) := \{\sigma \in G : x^\sigma = y\}$.
Recall that $\mathrm{ISO}_G(x, y)$ is either the empty set or a coset $\mathrm{Aut}_G(x)\sigma$. The coset is represented by an isomorphism $\sigma \in G$ (so $x^\sigma = y$) and a list of generators of $\mathrm{Aut}_G(x)$.

## 12.1 Representation issues in Luks's Algorithm

Descent
If $H \leq G$, then $G$-isomorphism reduces to $|G : H|$ instances of $H$-isomorphism, using the following equation. Write $G = \bigsqcup_{a \in R} Ha$ as the disjoint union of cosets of $H$, where $R$ is a set of coset representatives.

$$\mathrm{ISO}_G(x, y) = \bigsqcup_{a \in R} \mathrm{ISO}_{Ha}(x, y) = \bigsqcup_{a \in R} \mathrm{ISO}_H(x, y^{a^{-1}}) \cdot a$$

The LHS is the desired output of this step. While it is a disjoint union of represented cosets in $H$, the LHS cannot simply be represented by taking the union of the generators and coset representative that represent each coset. Recall that the membership problem for permutation groups can be solved in polynomial time.

**HW 12.1** (Combining subcosets to a coset)**.** Let $H \leq G$ and $R$ be a set of right coset representatives of $H$ in $G$. For each $a \in R$ let either $K_a \leq H$ or $K_a = \emptyset$. Assume the union $\bigsqcup_{a \in R} K_a a$ is a subcoset of $G$, i.e., $\bigsqcup_{a \in R} K_a a = Lb$ for some $L \leq G$ and $b \in G$. Given $R$ and the $K_a$ ($a \in R$), compute the coset $Lb$ in polynomial time. (Note: complexity is always relative to the size of the input, so "polynomial time" in this problem means time $\mathrm{poly}(n, |R|)$.

**Theorem 12.2** (Pálfy-Wolf (1982))**.** *If $G \leq S_n$ is primitive and solvable, then $|G| < n^C$, for $C = 3.24399\ldots$.*

**Theorem 12.3** (Luks (1980) + Pálfy-Wolf (1982))**.** *If $G$ is solvable, then* GI *can be solved in polynomial time.*

**Remark 12.4.** Let $\mathcal{C}$ be a class of groups closed under (1) subgroups and (2) quotients. If $G \in \mathcal{C}$, then all groups encountered by the Luks reduction also belongs to $\mathcal{C}$.

## 12.2 Examples of large primitive subgroups of $S_n$

The **giants** are $S_n$ and $A_n$.

The **Johnson groups**[1] are $S_k^{(t)}$ and $A_k^{(t)}$, the induced $S_k$ and $A_k$ action on $t$-tuples $\binom{[n]}{t}$ in $[n]$. The order is

$$|S_k^{(t)}| = k! \approx \exp(k) \approx \exp(n^{1/t}).$$

(Here the "approximate equality" sign is used informally; lower-order error terms occur in the exponents.)

Let $n = q^d$. The **affine linear group** $\mathrm{AGL}(d,q) := \{x \mapsto Ax + b : A \in \mathrm{GL}(d,q), b \in \mathbb{F}_q^d\}$ acts on $\mathbb{F}_q^d$ as affine linear transformations. The order is $|\mathrm{AGL}(d,q)| = \Theta(q^{d^2+d}) = \Theta(n^{d+1})$.

(Recall the $\Theta$ notation: if $a_n, b_n$ are sequences of positive numbers then we write $a_n = \Theta(b_n)$ if $a_n = O(b_n)$ and $b_n = O(a_n)$, i.e., there exist positive constants $c, C$ such that for all sufficiently large $n$ we have $ca_n \le b_n \le Ca_n$.)

**Theorem 12.5** (CFSG)**.** *Classification of finite simple groups, a sketch.*

- *Cyclic groups $\mathbb{Z}_p$.*
- *Alternating groups $A_n$, for $n \ge 5$.*
- *Lie-type simple groups. These are matrix groups over finite fields. They split into classical groups and exceptional groups.*
  - *Classical groups: linear, symplectic, orthogonal (3 types), and unitary groups. These are all parametrized by $(d,q)$.*
  - *Exceptional groups: about 10 classes. These are parametrized by $q$ (so $d$ is fixed).*
- *Sporadic groups. These have bounded order $|G| \le C$.*

## 12.3 Babai-Cameron-Pálfy

We discuss a bound on the order of primitive permutation groups in terms of the degree of the largest alternating section, made precise below.

**Definition 12.6.** $H$ is a **section** of $G$ (or $H$ is **involved** in $G$) if $H$ is a quotient of a subgroup of $G$.

**Theorem 12.7** (Babai-Cameron-Pálfy)**.** *Fix a constant $C$. Let $G \le S_n$. If $G$ is primitive and no composition factor of $G$ is either $A_k$ for $k > C$ or a classical group of dimension $> C$, then $|G| < n^c$, where $c$ depends on $C$.*

The conditions of the Babai-Cameron-Pálfy theorem are equivalent to the condition that $G \le S_n$ does not involve $A_k$ for $k > C$. This holds because $A_k \le \mathrm{SL}(k,q)$. A similar situation holds for other classical groups under a multiplicative factor to $C$.

**Definition 12.8.** We say that the **thickness** of $G$, denoted by $\Theta(G)$, is the largest $k$ such that $A_k$ is involved in $G$.

[1] This terminology is not standard, but is used by the instructor as they describe the automorphism groups of the Johnson graphs.

The following stronger version of BCP was proved by Pyber and Liebeck–Shalev.

**Theorem 12.9** (Strong version of BCP)**.** *If $G \leq S_n$ is primitive, then $|G| < n^{O(\Theta(G))}$.*

**Corollary 12.10.** StringIso *can be solved in time $n^{O(\Theta(G))}$.*

## 12.4 Cook reduction

Let $L_1$ and $L_2$ be languages, $L_i \subseteq \Delta_i^*$. A **Cook reduction** of $L_1$ to $L_2$ is a polynomial-time algorithm that takes as input a string $x \in \Delta_1^*$ and decides membership in $L_1$ by, time to time, querying an "oracle" for membership in $L_2$.

The **oracle** is a black box that takes an input and produces an output at no cost. So a Cook reduction will, time to time, compute strings $y_i \in \Delta_2^*$ and "magically" receive an answer to the question $y_i \stackrel{?}{\in} L_2$; this answer can then be used in the rest of the computation, including in computing additional srings to query.

We say that $L_1$ is **Cook reducible** to $L_2$ (notation: $L_1 \propto_{\text{Cook}} L_2$) if a Cook reduction of $L_1$ to $L_2$ exists.

**DO 12.11.** If $L_1 \propto_{\text{Cook}} L_2$ and $L_2 \in \text{P}$ (membership in $L_2$ can be decided in polynomial time) then $L_1 \in \text{P}$. (Hint: simulate the oracle by a subroutine that tests membership in $L_2$.)

**DO 12.12.** If $L_1 \propto_{\text{Cook}} L_2$ and $L_2 \in \text{NP} \cap \text{coNP}$ (membership in $L_2$ is "well characterized") then $L_1 \in \text{NP} \cap \text{coNP}$.

While Karp reducibility only applies to *decision problems* (where the output is binary, such as yes/no, true/false, 1/0, member/not member, etc.), the concept of Cook reducibility applies to computation problems (where the output is an arbitrary string). Below we give a more accurate definition of Cook reduction, in the multitape Turing machine model.

**Definition 12.13.** Let $\Delta_i, \Phi_i$ be alphabets for $i = 1, 2$. Let $g_i : \Delta_i^* \to \Phi_i^*$. A **Cook-reduction** of $g_1$ to $g_2$ is a polynomial-time oracle Turing machine which under a $g_2$-oracle computes $g_1$.

We say that the function $g_1$ is **Cook-reducible** to $g_2$ if there exists a Cook reduction from $g_1$ to $g_2$.

Cook reduction of languages is a special case: take $g_i$ to be the charcateristic function (membership indicator) of $L_i$.

We explain the concept of an oracle Turing machine.

The machine has a read-only input tape and a forward-only output tape; and it has a fixed number of read-write worktapes. One of the worktapes is designated the "query tape," another the "oracle tape." A subset of states of the finite control is designated "query states." When the machine enters a query state, the oracle evaluates the content of the query tape, $t$, and (magically) prints the value $g_2(t)$ on the "oracle tape" (erasing its previous content).

Note that part of the requirement is that the outputs to our queries should never be too long, they all must be of length, polynomially bounded in the length of the original input.

If we wanted to use a Cook reduction to actually compute $g_1$, we would need to replace the oracle calls by calls to a subroutine that computes $g_2$.

**DO 12.14.** Isomorphism of trivalent graphs $\propto_{\text{Cook}}$ Isomorphism of connected trivalent graphs.

If $k$ is the number of connected components, then $\leq \binom{k+1}{2}$ oracle calls are required.

## 12.5 Trivalent graph isomorphism

**Theorem 12.15** (Luks)**.** *Trivalent graph isomorphism is computable in polynomial time.*

We make some preliminary observations.

**HW 12.16** (Individualizing an edge)**.** "Isomorphism of connected trivalent graphs" $\propto_{\text{Cook}}$ "Isomorphism of connected trivalent graphs with one special edge."

**HW 12.17.** GraphIso $\propto_{\text{Karp}}$ Connected GraphIso.

**HW 12.18.** GraphIso $\propto_{\text{Karp}}$ Bipartite GraphIso.