Graph Isomorphism course, Spring 2017
Instructor: László Babai
Notes by Angela Wu and instructor
Tuesday, May 2, 2017

11 Day 11, TWk6

11.1 StringIso, definition

We define String Isomorphism (StringIso), which is intuitively the problem of deciding whether two strings are anagrams under a given group action.

First some notation. We denote by Σ a finite alphabet and by Ω the set of 'positions' in a "string." A string $x \in \Sigma^\Omega$ is a function $x : \Omega \to \Sigma$. For $\sigma \in \mathrm{Sym}(\Omega)$ and $x \in \Sigma^\Omega$, σ acts on x in a natural way. We write x^σ to represent the string $(x^\sigma)_i = x_{i^{\sigma^{-1}}}$. This gives an induced action $\mathrm{Sym}(\Omega) \to \mathrm{Sym}(\Sigma^\Omega)$.

Let $M \subseteq \mathrm{Sym}(\Omega)$. We say that two strings $x, y \in \Omega^\Sigma$ are M**-isomorphic** ($x \cong_M y$) if

$$(\exists \sigma \in M)(x^\sigma = y).$$

String Isomorphism (StringIso)
Input: strings $x, y : \Omega \to \Sigma$ and a group $G \le \mathrm{Sym}(\Omega)$ (given by a list of generators)
Question: Is $x \cong_G y$?

11.2 Decision problems, Karp-reducibility

We use the word "classical string" to mean a string where the set of positions is $[n]$ (where n is the length of the string). For a language $L \subseteq \Delta^*$ (all classical strings over the alphabet Δ), the **decision problem associated with** L is the *membership problem:* given $x \in \Delta^*$, is $x \in L$?

Consider languages $L_i \subseteq \Delta_i^*$ $(i - 1, 2)$. A **Karp reduction** from L_1 to L_2, is a function $f : \Delta_1^* \to \Delta_2^*$ such that

(1) f is polynomial-time computable, and

(2) $(\forall x \in \Delta_1^*)(x \in L_1 \iff f(x) \in L_2)$.

We say that L_1 is Karp-reducible to L_2, denoted $L_1 \propto_{\text{Karp}} L_2$, if such an f exists.

Theorem 11.1 (Cook–Levin, 1972)**.** *Every language in* NP *is Karp-reducible to* 3-Sat.

11.3 GI is Karp-reducible to StringIso

Proposition 11.2 (Luks)**.** GI $\propto_{\text{Karp}}$ StringIso.[1]

We describe the Karp reduction.

Denote by *code*(X) the $(0,1)$-string of length $\binom{n}{2}$ encoding the adjacency relation of X. ($\Omega = \binom{[n]}{2}$ and $\Sigma = \{0, 1\}$.) The group G for StringIso is $S_n^{(2)}$, the induced action of S_n on the $\binom{n}{2}$ pairs. (So $S_n \cong S_n^{(2)} \le S_{\binom{n}{2}}$.)

[1] View GI and StringIso as languages.

DO 11.3. Let X, Y be graphs on n vertices. Then $X \cong Y$ if and only if $code(X) \cong_{S_n^{(2)}} code(Y)$.

Now the Karp-reduction is the function

$$f(X,Y) = (code(X), code(Y), S_n^{(2)})$$

where n is the number of vertices of X and Y.

To be more precise, the Karp-reduction is supposed to also be defined for pairs (X, Y) of graphs which do not have the same number of vertices. Such pairs of course are never isomorphic.

DO 11.4. Extend the definition of f to such pairs of graphs.

11.4 Luks's theorem

Theorem 11.5 (Luks (1980))**.** GI *of graphs of bounded degree can be tested in polynomial time.*

A special case is when the degrees of this graph ≤ 3. This is derived from StringIso for 2-groups.

HW 11.6 (Tutte (1947))**.** Let X be a connected graph of degree ≤ 3. Let e be an edge in X. Show that $(\mathrm{Aut}(X))_e$ is a 2-group.

For the special case of p-groups G, StringIso under G is solvable in polynomial time. We spend the rest of the lecture giving a full proof of this result.

Theorem 11.7 (Luks)**.** SI *is solvable in polynomial time for p-groups G.*

► Notation.

Definition 11.8. For graphs X, Y, we denote

$$\mathrm{ISO}(X,Y) = \{f : X \to Y \quad | \quad f \text{ is a graph isomorphism } \}.$$

Definition 11.9. For strings $x, y \in \Sigma^\Omega$ and $M \subseteq \mathrm{Sym}(\Omega)$, we denote

$$\mathrm{ISO}_M(x,y) := \{\sigma \in M \quad | \quad x^\sigma = y\}.$$

DO 11.10. For graphs X, Y, the set $\mathrm{ISO}(X, Y)$ is either empty or a right coset of $\mathrm{Aut}(X)$. Namely, if $\sigma \in \mathrm{ISO}(X, Y)$, then $\mathrm{ISO}(X, Y) = \mathrm{Aut}(X) \cdot \sigma$.

DO 11.11. For strings $x, y \in \Sigma^\Omega$ and a group $G \leq \mathrm{Sym}(\Omega)$ we have $\mathrm{ISO}_G(x,y) = \begin{cases} \emptyset & x \ncong_G y \\ \mathrm{Aut}_G(x)\sigma & x \cong_G y \end{cases}$, where σ is any element in $\mathrm{ISO}_G(x, y)$.

DO 11.12. Let $G \leq \mathrm{Sym}(\Omega)$, $\sigma \in \mathrm{Sym}(\Omega)$, and $x, y \in \Sigma^\Omega$. Then,

$$\mathrm{ISO}_{G\sigma}(x,y) = \{\tau \in G\sigma : x^\tau = y\} = \mathrm{ISO}_G(x, y^{\sigma^{-1}})\sigma.$$

11.5 Luks's group theoretic Divide-and-Conquer method for SI

Luks's method basically combines two tricks. We call them (1) the "Chain Rule" (window-by-window processing, see velow), and "descent" (to a subgroup).

11.5.1 Descent

(2) Descent: Let $H \leq G$. Let R be a set of right coset representatives of H in G, so that $G = \bigcup_{a \in R} Ha$. Then,

$$\mathrm{ISO}_G(x, y) = \bigcup_{a \in R} \mathrm{ISO}_{Ha}(x, y).$$

Thus, ISO_G reduces to $|G : H|$ instances of ISO_H.

11.5.2 Chain Rule

Let $W \subseteq \Omega$ (the "window"). Define the "partial string" x^W (what we "see" through the window) by

$$(x^W)_i = \begin{cases} x_i & i \in W \\ * & \text{otherwise} \end{cases}$$

(where $*$ is a special symbol, not in the alphabet Σ).

Assume now that W is invariant under $M \subseteq \mathrm{Sym}(\Omega)$. Denote by

$$\mathrm{ISO}_M^W(x, y) := \mathrm{ISO}_M(x^W, y^W). \tag{1}$$

Intuitively, we restrict to W and solve on this smaller set. Let us write $\mathrm{ISO}_{M|_W}(x, y)$ to denote the set $\mathrm{ISO}_{M|_W}(x|_W, y|_W)$ (everything is restricted to the window). While computing $\mathrm{ISO}_{M|_W}(x, y)$ (by a recursive call to the smaller domain W), we need to keep track of the "tails" (action outside the window) of the group elements computed so we shall be able to interpret the result as a subset of $\mathrm{Sym}(\Omega)$.

Let now M be a group: $M = G \leq \mathrm{Sym}(\Omega)$. Consider the projection $\pi : G \to G|_W$. This in particular maps $\mathrm{ISO}_G^W(x, y)$ onto $\mathrm{ISO}_{G|_W}(x, y)$. Once we have found this image (by our recursive call), we need to lift this coset back to G.

Lifting the generators of $\mathrm{Aut}_{G|_W}(x)$ along with a coset representative is not sufficient. We also need to find $\mathrm{Ker}(\pi)$ (see the following DO exercise).

DO 11.13. If $\phi : G \to H = \langle t_1, \ldots, t_k \rangle$, let $s_1, \ldots, s_k \in G$ be such that $\phi(s_i) = t_i$. Then, $G = \langle s_1, \ldots, s_k, \mathrm{Ker}(\phi) \rangle$.

Now $\mathrm{Ker}(\pi) = G_{(W)}$, the pointwise stabilizer of the window.

DO 11.14. Given $G \leq \mathrm{Sym}(\Omega)$ (as always, by a list of generators), and $W \subseteq \Omega$, compute the pointwise stabilizer $G_{(W)}$ in polynomial time.

▶ Proceeding window-by-window

(1) Chain Rule: Let $G \leq \mathrm{Sym}(\Omega)$ and $\sigma \in \mathrm{Sym}(\Omega)$. Write $\Omega = \Omega_1 \sqcup \ldots \sqcup \Omega_k$ as a disjoint union of subsets invariant under G and σ (i.e., invariant under the group $\langle G, \sigma \rangle$, i.e., each Ω_i is a union of orbits of the group $\langle G, \sigma \rangle$).

We compute $\mathrm{ISO}_{G\sigma}(x, y)$ progressing sequentially through the windows.

Procedure Chain Rule

Input: $x, y, G, \sigma, \Omega = \Omega_1 \sqcup \ldots \sqcup \Omega_k$

Initialize $M \leftarrow G\sigma$
For $i = 1 \ldots k$
  $M \leftarrow \mathrm{ISO}_M^{\Omega_i}(x, y)$
Return M

Note the following loop invariant: M is either empty or a subcoset of $G\sigma$. (True at the beginning and remains true under each iteration of the "for" loop.)

DO 11.15. Prove: the procedure outputs $\mathrm{ISO}_{G\sigma}(x, y)$.

11.6 Divide-and-Conquer: combining Descent and the Chain Rule

We describe Luks's strategy for the SI problem.

If G is intransitive, process orbit by orbit via the Chain rule.

If G is transitive but imprimitive, let $\mathcal{B} = \{B_1, \ldots, B_k\}$ be a minimal system of imprimitivity (a G-invariant partition into maximal blocks).

So, $\Omega = \bigsqcup B_i$ and the B_i are maximal blocks. We have a G-action $G \curvearrowright \mathcal{B}$ (the group G permutes the blocks). This means an action $\phi : G \to S_k$. Let $\widetilde{G} = \mathrm{Img}(\phi)$ and $K = \mathrm{Ker}(\phi)$.

DO 11.16. Prove $\widetilde{G}$ is a primitive group (because the blocks are maximal).

Having run out of simple ways to "divide," the naive implementation of Luks's method is to do exhaustive search on $\widetilde{G}$, i.e., to descend to the kernel K.

We shall see that in important cases this already yields a polynomial-time algorithm.

11.7 Complexity estimation

11.7.1 Chain Rule: ultra-efficient recurrence

Let $f(n)$ denote the cost (number of group operations performed) on instances with domain size n in the worst case. We assume we have pruned the set of generators, so G is given by a list of $\leq 2n$ generators.

Write $|\Omega_i| = n_i$. We notice that $f(n) \leq \sum f(n_i) + n^c$ where we write n^c for the (polynomial-time) cost of the overhead (bookkeeping, and putting together the pieces received from the recursive calls – all that is polynomial-time). Then, $f(n) \leq n^{c+1}$.

Justification: Evaluation of recurrences by **the method of reverse inequalities**.
Suppose that $g(n) \geq f(n)$ for $n \leq n_0$ and $g(n) \geq \sum g(n_i) + n^c$ for $n > n_0$.
Then, by induction, $f(n) \leq g(n)$.
So all we need to do is guess a function g and a threshold n_0 such that g satisfies the reverse inequality above the threshold. Guess $g(n) := n^{c+1}$ and $n_0 = 1$.

11.7.2 Analyzing Luks's strategy for SI

The chain rule very efficiently reduces the problem to transitive G. We analyze the case when G is transitive. Recall that k denotes the number of blocks in our minimal system of imprimitivity. We reduce to K (the kernel of the action on the set of blocks). Each block is K-invariant, so we proceed block by block using the Chain Rule. The cost of the second phase is $\le k \cdot f(n/k)$, and we need to repeat this for every coset of K in G (cost of descent), so our overall estimate is

$$f(n) \le k \cdot f(n/k) \cdot |G : K|, \text{ where } |G : K| = |\widetilde{G}|.$$

Let us now consider the case when G is a p-group.

Lemma 11.17. *If $G \le S_n$ is a primitive p-group then $|G| = n = p$.*

To prove this, we make some observations.

DO 11.18. If G is a transitive p-group then $n = p^\ell$ for some ℓ. (Hint: orbit-stabilizer lemma.)

So $G \le P$ for some $P \in \mathrm{Syl}_p(S_{p^\ell})$. Therefore any system of imprimitivity of P is also a system of imprimitivity of G.

DO 11.19. Infer the Lemma from these observations and the structure of the Sylow p-subgroups of S_{p^ℓ}.

So if G is a p-group then we have $k = |\widetilde{G}| = p$ and therefore our recurrence (disregarding the overhead) becomes $f(p^\ell) \le p \cdot f(p^{\ell-1}) \cdot p = p^2 \cdot f(p^{\ell-1})$. Thus, $f(p^\ell) = O(p^{2\ell})$, so $f(n) = O(n^2)$.

We have thus completed the proof of Theorem 11.7 which we restate here.

Theorem 11.20 (Luks)**.** StringIso *for p-groups G can be solved in polynomial time. (The polynomial does not depend on the prime p.)*

The following group-theoretic result, proved independently and simultaneously by T. R. Wolf and the instructor's student Péter P. Pálfy (nicknamed p^3), allows us to extend the theorem to all solvable groups.

Theorem 11.21 (Pálfy–Wolf, 1982)**.** *If $G \le S_n$ is solvable and primitive, then $|G| \le n^C$, where $C = 3.24399\ldots$.*

HW 11.22. Combine Luks's method with the Pálfy–Wolf Theorem to show that StringIso for solvable G can be solved in polynomial time.