# Algorithms for permutation groups
# Part I

Adéla Skoková

March 25, 2012

# Content

- Runing times for algorithms for permutation groups
- Basic definitions
- The Sifting procedure
- Schreier's Lemma

# Basic notation

A permutation group **G** is a group whose elements are permutations of the given set $\Omega$.

Group of all permutations is symetric group $Sym(\Omega)$ and group of permutations is its subgroup.

Group operations are composition of permutations in **G**.

Suppose that $|\Omega| = n$.
We can identify $\Omega$ with $\{1, 2, 3, ..., n\}$.

# Owerview of permutation group algorithm

Input to the algorithm which works with permutation group: list of generators of the group.

We have given $\mathbf{G} = \langle \mathbf{S} \rangle \leq \mathbf{S}_n$, the input lenght can be $|\mathbf{S}|n$.
A polynomial-time algorithm – $O((|\mathbf{S}|n)^c)$ for some fixed $c$.
In practice $|\mathbf{S}|$ is usually small.

Experience shows that a lot of ideas, developed in the polynomial-time context, are later incorporated in practical algorithms; conversely, procedures performing well in practice often have versions with polynomial running time.

## Some tasks - deterministic polynomial-time algorithm

- given $h \in Sym(\Omega)$, test whether $h \in \mathbf{G}$
- find the order of $\mathbf{G}$
- find orbits, center or blocks of imprimitivity of $\mathbf{G}$
- ...

## Definition (Small-base group)

We call an (infinite) family $\Delta$ of permutation groups small-base groups if each $\mathbf{G} \in \Delta$ of degree $n$ satisfies $\log |\mathbf{G}| < \log^c n$ for some fixed constant $c$.

Example: primitive groups not containing alternating composition.

# Nearly-linear time algorithms

The nearly linear time, $O(n|\mathbf{S}|\log^{c'}(n|\mathbf{S}|))$, of the input length.

The time bound of nearly linear-time algorithms on small-base input groups is $O\left(n|\mathbf{S}|\right)$.

## Non-polynomial-time methods

- Given $\Delta \subseteq \Omega$, compute the setwise stabilizer $G_\Delta = \{g \in \mathbf{G} | \Delta^g = \Delta\}$.
- Given $\mathbf{H}, \mathbf{G} \leq Sym(\Omega)$, compute $\mathbf{C_G}(\mathbf{H})$ *centralizer*.
- Given $\mathbf{H}, \mathbf{G} \leq Sym(\Omega)$, compute $\mathbf{G} \cap \mathbf{H}$.
- Given $x_1, x_2 \in \mathbf{G}$, decide whether they are conjugate.

It is concievable that there may be polynomial time algorithms (at least for the classes of groups occurring in practice) to solve them.

# Basic definitions

## Definition

A sequence of elements $\mathbf{B} = (\beta_1, ..., \beta_m)$ from $\Omega$ is called a base for $\mathbf{G}$ if the only element of $\mathbf{G}$ to fix $\mathbf{B}$ pointwise is the identity.

The sequence $\mathbf{B}$ defines a subgroup chain

$$\mathbf{G} = \mathbf{G}^{[1]} \leq \mathbf{G}^{[2]} \leq \ldots \leq \mathbf{G}^{[m]} \leq \mathbf{G}^{[m+1]} = 1,$$

where $\mathbf{G}^{[i]} := \mathbf{G}_{(\beta_1,\ldots,\beta_{i-1})}$ is the pointwise stabilizer of $\{\beta_1, \ldots, \beta_{i-1}\}$.

## Definition

The base is called nonredundant if $\mathbf{G}^{[i+1]}$ is a proper subgroup of $\mathbf{G}^{[i]}$ for all $i \in [1, m]$.

Different nonredundant bases can have different size.

# A small-base group

Repeating Lagranges theorem:

$$|\mathbf{G}| = \prod_{i=1}^{m} |\mathbf{G}^{[i]} : \mathbf{G}^{[i+1]}|.$$

The cosets of $\mathbf{G}^{[i]}$ mod $\mathbf{G}^{[i+1]}$ correspond to the elements of the orbit $\beta_i^{\mathbf{G}^{[i]}}$, we obtain $|\mathbf{G}^{[i]} : \mathbf{G}^{[i+1]}| = |\beta_i^{\mathbf{G}^{[i]}}| \leq n$ for all $i \in [1, m]$.

If $\mathbf{B}$ is nonredundant then $|\mathbf{G}^{[i]} : \mathbf{G}^{[i+1]}| \geq 2$.

$$2^{|\mathbf{B}|} \leq |\mathbf{G}| \leq n^{|\mathbf{B}|}$$

$$\frac{\log |\mathbf{G}|}{\log n} \leq |\mathbf{B}| \leq \log |\mathbf{G}|$$

The last inequality justifies the name 'small-base group'.

# Strong generating set

## Definition

A strong generating set (SGS) for $\mathbf{G}$ relative to $\mathbf{B}$ is a generating set $\mathbf{S}$ for $\mathbf{G}$ with the property that

$$\left\langle \mathbf{S} \cap \mathbf{G}^{[i]} \right\rangle = \mathbf{G}^{[i]}, \text{ for } 1 \leq i \leq m+1.$$

# Example

A group $\mathbf{G} = \mathbf{S}_4$ in its natural action on the set $[1,4] = \{1,2,3,4\}$.
$\mathbf{B} = (1,2,3)$ is nonredundant base for $\mathbf{G}$.

$\mathbf{G}^{[1]} = Sym([1,4]) \lneq \mathbf{G}^{[2]} = Sym([2,4]) \lneq \mathbf{G}^{[3]} = Sym([3,4]) \lneq \mathbf{G}^{[4]} = 1$

$\mathbf{S} = \{(1,2,3,4),(3,4)\}$ is not strong generating set relative to $\mathbf{B}$ since $\left\langle \mathbf{S} \cap \mathbf{G}^{[2]} \right\rangle = Sym([3,4]) \neq \mathbf{G}^{[2]} = Sym([2,4])$.

$\mathbf{T} = \{(1,2,3,4),(2,3,4),(3,4)\}$ is an SGS relative to $\mathbf{B}$.

# Fundamental orbits

**Definition**

Orbits $\beta_i^{\mathbf{G}^{[i]}}$ of SGS are called fundamental orbits of $\mathbf{G}$.

By $|\mathbf{G}| = \prod_{i=1}^{m} |\mathbf{G}^{[i]} : \mathbf{G}^{[i+1]}|$ we can see that $|\mathbf{G}| = \prod_{i=1}^{m} |\beta_i^{\mathbf{G}^{[i]}}|$.

Given SGS, the orbits $\beta_i^{\mathbf{G}^{[i]}}$ can be computed easily.

Keeping track of elements of $\mathbf{G}^{[i]}$ in the orbit algorithm that carry $\beta_i$ to points in $\beta_i^{\mathbf{G}^{[i]}}$, we obtain transversals $R_i$ for $\mathbf{G}^{[i]} \mod \mathbf{G}^{[i+1]}$

# The Sifting Procedure

Every $g \in \mathbf{G}$ can be written uniquely in the form $g = r_m r_{m-1} \dots r_1$ with $r_i \in R_i$, (Lagrange's theorem).

This decomposition can be done algorithmically:
Given $g \in \mathbf{G}$, find the coset representative $r_1 \in R_1$ such that $\beta_1^g = \beta_1^{r_1}$.

Then compute $g_2 := g r_1^{-1} \in \mathbf{G}^{[2]}$; find $r_2 \in R_2$ such that $\beta_2^{g_2} = \beta_2^{r_2}$;
compute $g_3 := g_2 r_2^{-1} \in \mathbf{G}^{[3]}$;
etc.

# Testing membership

Given $h \in Sym(\Omega)$,
We can try to factor $h$ as a product of coset representatives.
Successful: $h \in \mathbf{G}$.

- for some $i \leq m$, the ratio $h_i := hr_1^{-1} r_2^{-1} \dots r_{i-1}^{-1}$ computed by the sifting procedure carries $\beta_i$ out of the orbit $\beta_i^{G[i]}$;
- $h_{m+1} := hr_1^{-1} r_2^{-1} \dots r_{m-1}^{-1} r_m^{-1} \neq 1$.

**Definition**

The ratio $h_i$ with the largest index $i$ $(i \leq m+1)$ computed by the sifting procedure is called the siftee of $h$.

# Schreier tree

## Definition

A Schreier tree data structure for $\mathbf{G}$ is a sequence of pairs $(S_i, T_i)$ called Schreier trees, one for each base point $\beta_i$, $1 \leq i \leq m$.

$T_i$ is a directed labeled tree, with all edges directed toward the root $\beta_i$ and edge labels from a set $S_i \subseteq \mathbf{G}^{[i]}$.

Vertices of $T_i$ are points of the fundamental orbit $\beta_i^{\mathbf{G}^{[i]}}$.

# Schreier tree II

Labels satisfy the condition that for each directed edge from $\gamma$ to $\delta$ with label $h$, $\gamma^h = \delta$.

If $\gamma$ is a vertex of $T_i$ then the sequence of the edge labels along the unique path from $\gamma$ to $\beta_i$ in $T_i$ is a word in the elements of $S_i$ such that the product of these permutations moves $\gamma$ to $\beta_i$.

Thus each Schreier tree $(S_i, T_i)$ defines inverses of a set of coset representatives for $G^{[i+1]}$ in $G^{[i]}$.

We store inverses of coset representatives in the Schreier trees because sifting requires the inverses of these transversal elements.

# Memory requirements

Memory requirement for storage:
$\mathbf{S}_i$ is $O(|\mathbf{S}_i|n)$
$\mathbf{T}_i$ is $O(n)$.

$T_i$ can be stored in an array $V_i$ of lenght $n$.

$\gamma$-th entry of $V_i$ is deffined iff $\gamma \in \beta_i^{\mathbf{G}^{[i]}}$.

$V_i[\gamma]$ is a pointer to the element of $S_i$.
It is the label of the unique edge of $T_i$ starting at $\gamma$.

# Example *continue*

$\mathbf{G} = \mathbf{S}_4$ with base $\mathbf{B} = (1, 2, 3)$
and SGS $\mathbf{T} = \{(1, 2, 3, 4), (2, 3, 4), (3, 4)\}$.

Construction of Schreier trees for $\mathbf{G}$ using label set $S_i := \mathbf{T} \cap \mathbf{G}^{[i]}$.

The trees $T_i$ can be constructed as the breadth-first-search trees, which compute the orbits $\beta_i^{\mathbf{G}^{[i]}}$.

The edges of the trees must be directed toward the roots, we have to use the inverses of the elements of $S_i$ in the construction of the $T_i$.

The label set $S_i$ determines uniquely only the levels of the tree $T_i$, because the vertices on level $j$ may be the images of more vertices on level $j - 1$, under more permutations.

# Example *continue*

Construction of Schreier trees for $\mathbf{G}$ using label set $S_i := \mathbf{T} \cap \mathbf{G}^{[i]}$.
In $T_1$:

- level 0 contains the point 1
- level 1 contains only the point 4 - it is the only point that is the image of 1 under the inverse of some element of $S_1$, and $(1,2,3,4)$ is the only possible label for the edge $\overrightarrow{(4,1)}$.
- level 2 contains only the point 3 - we have three possibilities for defining the label of $\overrightarrow{(3,4)}$ because the inverses of $(1,2,3,4)$, $(2,3,4)$, and $(3,4)$ all map 4 to 3.

The labels of $\overrightarrow{(3,4)}$ depends on the order of the elements $S_1$.

## Example *continue*

One possibility for Schreier tree:

$$(id, (2,3,4), (2,3,4), (1,2,3,4)),$$

$$(*, id, (2,3,4), (2,3,4)),$$

$$and(*, *, id, (3,4)),$$

here $*$ denotes that the appropriate entry of the array is not defined because the corresponding point is not in the fundamental orbit of $\beta_i$.

# Example *continue*

A transversal element carrying the first base point 1 to 3. From the first array we obtain that:

$$(2,3,4).(1,2,3,4) = (1,2,4,3)$$

maps 3 to 1.

Its inverse is the desired transversal element.

# Schreier's Lemma

**Lemma (Schreier's Lemma)**

*Let $\mathbf{H} \leq \mathbf{G} = \langle \mathbf{S} \rangle$ and let $\mathbf{R}$ be a right transversal for $\mathbf{G}$ mod $\mathbf{H}$, with $1 \in \mathbf{R}$. Then the set*

$$\mathbf{T} = \left\{ rs(\overline{rs})^{-1} | r \in \mathbf{R}, s \in \mathbf{S} \right\}$$

*generates $\mathbf{H}$.*
*The elements of $\mathbf{T}$ are called Schreier generators for $\mathbf{H}$.*

$\overline{r}$ is the chosen representative in the transversal $\mathbf{R}$ of the coset $\mathbf{H}g$, that is $g \in \mathbf{H}\overline{g}$.

The lemma is used in the Schreier-Sims algorithm and also for finding a presentation of a subgroup.

# Schreier's Lemma - proof

**Proof.**

By definition, the elements of $\mathbf{T}$ are in $\mathbf{H}$,

it is enough to show that $\mathbf{T} \cup \mathbf{T}^{-1}$ generates $\mathbf{H}$.

$$\mathbf{T}^{-1} = \left\{ rs(\overline{rs})^{-1} | r \in \mathbf{R}, s \in \mathbf{S}^{-1} \right\}$$

$$\mathbf{T} = \left\{ rs(\overline{rs})^{-1} | r \in \mathbf{R}, s \in \mathbf{S} \right\}$$

Let $h \in \mathbf{H}$ be arbitrary.

Since $\mathbf{H} \leq \mathbf{G}$, $h = s_1 s_2 \dots s_k$ for $k \in \mathbb{N}$
and $s_i \in \mathbf{S} \cup \mathbf{S}^{-1}$ for $i \leq k$.

□

# Schreier's Lemma - proof

**Proof.**

We define a sequence $h_0, h_1, ..., h_k$ of group elements such that

$$h_j = t_1 t_2 \dots t_j r_{j+1} s_{j+1} s_{j+2} \dots s_k,$$

with $t_i \in \mathbf{T} \cup \mathbf{T}^{-1}$ for $i \leq j$, $r_{j+1} \in \mathbf{R}$, and $h_j = h$.

Let $h_0 := 1 s_1 s_2 \dots s_k$.

Recursively, if $h_j$ is already defined then let

$$t_{j+1} := r_{j+1} s_{j+1} (\overline{r_{j+1} s_{j+1}})^{-1}$$

$$r_{j+2} := r_{j+1} s_{j+1}.$$

Clearly, $h_{j+1} = h_j = h$, and it has the required form. $\square$

# Schreier's Lemma - proof

**Proof.**

We have $h = h_k = t_1 t_2 \dots t_k r_{k+1}$.

Since $h \in \mathbf{H}$ and $t_1 t_2 \dots t_k \in \langle \mathbf{T} \rangle \leq \mathbf{H}$,
we must have $r_{k+1} \in \mathbf{H} \cap \mathbf{R} = 1$.
Hence $h \in \langle \mathbf{T} \rangle$. $\square$ $\square$

# Remarque

We deal only with finite groups, and so every element $h$ of a given group $\mathbf{G} = \langle \mathbf{S} \rangle$ can be written as a product $h = s_1 s_2 \dots s_k$ of generators and we do not have to deal with the possibility that some $s_i$ is the inverse of a generator. In the proof of Lemma, we included the possibility $s_i \in \mathbf{S}^{-1}$ since this lemma is valid for infinite groups as well, and in an infinite group we may need the inverses of generators to write every group element as a finite product.

Thank you for your attention.