# Algorithmic Abstract Algebra 2
# Permutation Groups

Peter Mayr

Linz, October, 2013

Last time we saw:

1. Group actions
   - 1.1 Orbit algorithm
   - 1.2 Stabilizers, representatives (Schreier vectors)
2. Pseudo-random elements

Today we'll look at:

3. Permutation groups
   - 3.1 Stabilizer chains
   - 3.2 Sifting
   - 3.3 Schreier-Sims algorithm
   - 3.4 Backtrack

# Permutation groups

Let $G$ be a group of permutations on $\Omega$ given by generators.
We want to compute with $G$, e.g.

1. find $|G|$,
2. determine membership in $G$,

without enumerating all its elements.

# Stabilizer chains

Let $G \leq S_\Omega$, let $\beta_1, \ldots, \beta_m \in \Omega$ such that

$$\forall g \in G : \beta_1^g = \beta_1, \ldots, \beta_m^g = \beta_m \Rightarrow g = 1.$$

Then $B = (\beta_1, \ldots, \beta_m)$ is a **base** for $G$.
With $G^0 := G$ and $G^i = \mathrm{Stab}_{G^{i-1}}(\beta_i)$ we have a **stabilizer chain**

$$G = G^0 \geq G^1 \geq G^2 \geq \cdots G^m = 1.$$

## Remark

1. If $g$ and $h$ agree on a base, then $g = h$.
2. $|G| = \prod_{i=1}^{m} |G^{i-1} : G^i|$ by Lagrange's Theorem.
3. If $G^{i-1} \neq G^i$, then $|G^{i-1} : G^i| \geq 2$. So the length of a base can be bounded by $\log_2 |G|$.

4. The union $S$ of generators for $G^0, \ldots, G^{m-1}$ is a **strong generating set** for $B$, ie,

$$\langle S \cap G^i \rangle = G^i \text{ for } 0 \leq i \leq m.$$

(cf. basis in row echelon form for a subspace of $\mathbb{R}^n$.)

## Exercise

What are irredundant bases for $S_n, A_n$?

# Sim's idea

1. If $g \in G^{i-1}$, then $g = ab$ with $a \in G^i$ and $b$ a coset representative for $G^i$ in $G^{i-1}$.
2. So for any $g \in G$,

$$g = b_m b_{m-1} \cdots b_1 \qquad \text{normal form!!}$$

   with $b_i$ a coset representative for $G^i$ in $G^{i-1}$.
3. By $G^i = \mathrm{Stab}_{G^{i-1}}(\beta_i)$, every $b_i$ corresponds to a point $\beta_i^{b_i}$ in the orbit of $\beta_i$ under $G^{i-1}$.

# How to build a stabilizer chain

## First try

Given $G = \langle g_1, \ldots, g_k \rangle$ acting on $\Omega$

1. Pick $\beta_1 \in \Omega$ that is not fixed by some $g_1, \ldots, g_k$.
2. Compute the orbit of $\beta_1$ and generators for its stabilizer $G^1$ in $G^0 = G$ using the Orbit-Algorithm.
3. Iterate for $G^1$ until we reach a trivial stabilizer.

**Problem:** In each step, the number of Schreier generators is linear in the index of the stabilizer. So we have about $|G|$ generators in the last step (**exponential algorithm**).

## Instead

Build the stabilizer chain recursively and use the partial chain to remove redundant generators.

# Storing a stabilizer chain

In GAP each $G^i$ in the stabilizer chain is represented by a record with the following entries

- the generators of $G^i$,
- the orbit of $\beta_{i+1}$ under $G^i$,
- a corresponding transversal (ie., Schreier vector), and
- a pointer to the record of the stabilizer $G^{i+1} = \mathrm{Stab}_{G^i}(\beta_{i+1})$.

## Example

Let $G := A_4$ with base $B := [1, 2]$.
Then $G = G^0 = \langle (1,2,3), (2,3,4) \rangle$, $G^1 = \mathrm{Stab}_G(1) = \langle (2,3,4) \rangle$, $G^2 = \mathrm{Stab}_G(1,2) = \langle \rangle$.

```
rec(
  generators := [ (1,2,3), (2,3,4) ],
  orbit := [ 1, 2, 3, 4 ],
  transversal := [ (), (1,2,3), (1,3,2), (1,4,2) ],
  stabilizer := rec(
      generators := [ (2,3,4) ],
      orbit := [ 2, 3, 4 ],
      transversal := [ (), (2,3,4), (2,4,3) ],
      stabilizer := rec(
          generators := [  ] ) ) )
```

# Membership test

## Factorization algorithm

**Input** : A stabilizer chain $C$ for a group $G$ and $g \in G$

**Output**: Coset reps $[b_1, b_2, \ldots, b_m]$ such that $g = b_m b_{m-1} \ldots b_1$

```
1  L := [];
2  while C.generators <> [] do
3      β := C.orbit[1];                          /* base element */
4      δ := β^g;
5      r := C.transversal[δ];                    /* rep that maps β to δ */
6      g := gr^{-1};
7      Add r to L;
8      C := C.stabilizer ;                       /* one step down the chain */
9  end
10 return L
```

## Correctness

Since $\beta^r = \beta^g$, the new $g$ in line 6 stabilizes $\beta$. When the while-loop finishes, $g = 1$.

If $g \notin G$, then computing a factorization will fail. So we obtain a membership test.

## ElementTest / Sifting

**Input** : A stabilizer chain $C$ for a group $G$ and a permutation $g$
**Output**: A factorization if $g \in G$ and an error otherwise

```
1  L := [];
2  while C.generators <> [] do
3      β := C.orbit[1];
4      δ := β^g;
5      if C.transversal[δ] does not exist then
6          return not contained
7      end
8      r := C.transversal[δ];
9      g := gr^{-1};
10     Add r to L;
11     C := C.stabilizer
12 end
13 if g ≠ 1 then
14     return not contained
15 end
16 return L
```

The following algorithm builds a stabilizer chain $C$ (a global record) using a recursive function Extend.

## Recursive Schreier-Sims algorithm

**Input** : Generators $[g_1, \ldots, g_k]$ for a permutation group $G \leq S_n$
**Output**: A recursive data structure for a stabilizer chain $C$ for $G$

```
1 C := rec(generators:=[]);
2 for i ∈ [1,...,k] do
3     if ElementTest(C,g_i) fails then
4         Extend(C,g_i);
5     end
6 end
7 return C
```

## Extend(C,a)

```
1  if C.generators = [] then
2      β := one point moved by a ;        /* may be predefined */
3      C.orbit := [β]; C.transversal := [1];
4      C.stabilizer := rec(generators:=[]);
5  end
6  Add a to C.generators; T := C.transversal;
7  for δ ∈C.orbit do
8      for b ∈ C.generators do
9          γ := δ^b;
10         if γ ∉C.orbit then
11             Add γ to C.orbit;
12             Add T[δ] · b to T
13         else
14             s := T[δ]bT[γ]^{-1};
15             Extend(C.stabilizer, s);
16         end
17     end
18 end
```

## Correctness

1. For $G \leq S_n$, the stabilizer chain has $\leq n$ levels and the Schreier-Sims algorithm terminates.
2. When a call of $\mathit{Extend}(C, a)$ is completed, we have for the base point $\beta := C.\mathit{orbit}[1]$ that

$$\mathrm{Stab}_{\langle C.\mathit{generators}\rangle}(\beta) = \langle C.\mathit{stabilizer}.\mathit{generators}\rangle$$

   (and similarly for all levels below).
   Hence $C$ is a proper stabilizer chain for $\langle C.\mathit{generators}\rangle$
3. When the Schreier-Sims algorithm finishes, then we have at the top level $\langle C.\mathit{generators}\rangle = \langle g_1, \ldots, g_k\rangle$.

## Performance

1. If $G \leq S_n$, then the Schreier-Sims algorithm has complexity polynomial in $n$ (and even works in practice).
2. The main problem is the large number of Schreier generators on each level. Heuristically at least half of them are redundant: heuristic randomized algorithm (Leon, 1980), nearly linear time Monte Carlo algorithm (Babai, et al, 1991).
3. Results from a randomized algorithm can be verified, e.g., if $|G|$ is known, by using a presentation of $G$, by using a composition series, ...

# Applications of stabilizer chains

Having a strong generating system/stabilizer chain we can do the following efficiently:

1. Test whether $g \in G$.
2. Calculate $|G|$.
3. Normal closure with element test.
4. Compute derived series and central series (check solvability, nilpotence).
5. Enumerate $G$.
6. Obtain random elements with guaranteed equal distribution.

## Exercise

Describe an algorithm for 6.

# Non-polynomial-time methods

No polynomial time algorithms are known for the following:

1. Given $\Delta \subseteq [1..n]$, compute the **setwise stabilizer** $\{g \in G \mid \Delta^g = \Delta\}$.
2. Given $H, G \leq S_n$, compute the **centralizer** $C_G(H)$.
3. Given $H, G \leq S_n$, compute the **intersection** $H \cap G$.
4. Given $x_1, x_2 \in G$, decide whether they are **conjugate** in $G$.

These tasks are polynomially equivalent, and **graph isomorphism** checking can be reduced to them (Luks, 1993).

Assuming $\mathbf{P} \neq \mathbf{NP}$, checking graph isomorphism is conjectured to have intermediate complexity.

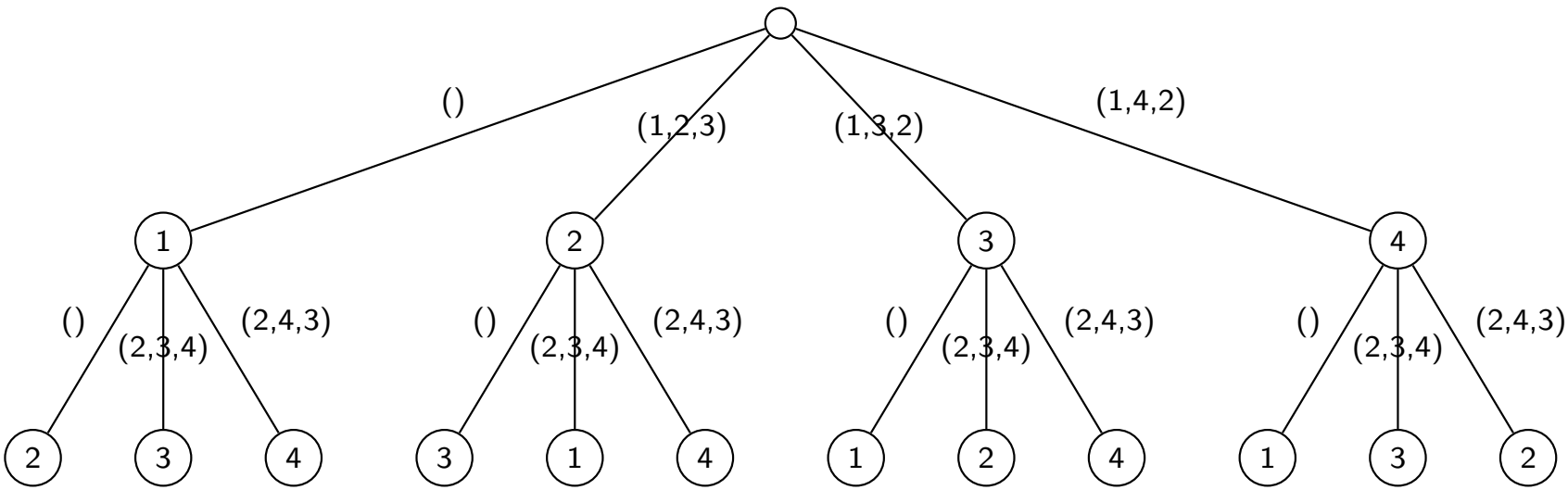
# If all else fails

Practical algorithms use **backtrack methods**:

1. **depth-first-search** of a tree of elements from a stabilizer chain to find elements satisfying the desired property.
2. depending on the property the search tree can be **pruned** so that not all group elements have to be enumerated
3. exponential worst case complexity but works well in practice.

# Search tree for $A_4$

- Vertices are images for the base points 1 and 2.
- Edges are labelled by transversal elements.
- Every leaf represents the group element that is the product of the edge labels on its path to the root.



# Example: Centralizer

We use backtracking for the tree on the previous slide to find the elements $c \in A_4$ that centralize $(1,2,3)$.

Note that $(1,2,3)c = c(1,2,3)$ implies $2^c = 1^{c(1,2,3)}$. So $c$ is uniquely determined by $1^c$.
This allows us to **prune** the search tree.

| $1^c$ | $2^c$ | $c$ |
|---|---|---|
| 1 | 2 | $()$ |
| 2 | 3 | $(1,2,3)$ |
| 3 | 1 | $(1,3,2)$ |
| 4 | 4 | $*$ |

Thus we enumerated all 3 elements in $C_{A_4}((1,2,3))$

# Some heuristics

Given a permutation group by generators, the following computational tasks are ordered from easiest to hardest:

1. orbits
2. stabilizer chain
3. chief series
4. conjugacy classes of elements
5. lattice of subgroups

# Example: Rubik's cube in GAP (Schönert, 1993)

`www.gap-systems.org/Doc/Examples/`

Enumerate the faces as follows and consider the subgroup of $S_{48}$ that is generated by 6 rotations.