Algorithmic Group Theory
Permutation Groups

Bettina Eick

TU Braunschweig

Douala, Cameroon, Oct 2017

Algorithmic Group Theory

General Introduction

Group representations

Algorithms for groups depend on how the group is given:

(1) Permutation groups: Subgroups of $Sym(n)$ given by generators.

(2) Matrix groups: Subgroups of $GL(n, R)$ given by generators.

(3) Finitely presented groups: $\langle X \mid R \rangle$.

(4) Polycyclic groups: Polycyclic presentations.

(5) Arithmetic groups: Subgroups of $GL(n, R)$ given by polynomials.

(6) ...

Permutation Groups

Cayley's theorem

Every finite group G embeds into $Sym(|G|)$.

Permutation Groups

(1) Algorithmic theory for permutation groups is highly developed.

(2) Most problems have efficient solutions.

(3) Seress: Permutation group algorithms.
Holt, Eick, O'Brien: Handbook of computational group theory.

Permutation Group Preliminaries

Element arithmetic

Element arithmetic

Elements of $Sym(n)$ are represented as product of disjoint cycles.

```
gap> a := (1,2,3);
(1,2,3)
gap> b := (2,3,4);
(2,3,4)
gap> a*b;
(1,3)(2,4)
```

Subgroups and generators

Generating sets

Subgroups are mostly defined by generating sets.

```
gap> G := Group( (1,2,3,4,5), (3,4,7) );
Group([ (1,2,3,4,5), (3,4,7) ])
gap> Size(G);
360
```

Efficient methods

Computing with permutation groups

Most problems have an efficient solution.

```
gap> a := Random(SymmetricGroup(100));;
gap> b := Random(SymmetricGroup(100));;
gap> Order(a);
16836
gap> Order(b);
4140
```

Efficient methods II

```
gap> G := Group(a,b);;
gap> Size(G);
933262154439441526816992388562667004907159682643816214685929
638952175999932299156089414639761565182862536979208272237582
511852109168640000000000000000000000
gap> time;
3185
```

Question

How can GAP determine the size of G so fast?

Orbits and Stabilizers

A fundamental algorithm

Orbit/Stabilizer

Given: $G = \langle g_1, \ldots, g_n \rangle \leq Sym(n)$ and $w \in \{1, \ldots, n\}$

> $W := [w]$; $T := [1_G]$; $S := []$; $i := 0$;

> while $i < Length(W)$ do

> > $i := i + 1$;

> > for $g \in \{g_1, \ldots, g_n\}$ do

> > > $v := w[i]^g$; $j := Position(W, v)$;

> > > if $j = false$ then $Add(W, v)$; $Add(T, T[i] * g)$; fi;

> > > if $j \neq false$ then $Add(S, T[i] * g * T[j]^{-}1)$; fi;

> return W, T and S;

A fundamental algorithm II

Orbit/Stabilizer

(1) Computes the orbit W and generators for the stabilizer S.

(2) Recall that $|W| = [G : S]$.

(3) An iterated computation allows to compute $|G|$.

(4) This is much faster than listing all elements.

Backtracking

Fundamental problems

Fundamental problems

Given $G = \langle g_1, \ldots, g_n \rangle \leq Sym(n)$.

(1) Given $g \in Sym(n)$, decide if $g \in G$.

(2) Given $g \in G$, compute $C_G(g)$.

(3) Given $U \leq G$, compute $N_G(U)$.

(4) Given $H \leq Sym(n)$, compute $G \cap H$.

(5) Compute the maximal subgroups of G.

(6) Compute the conjugacy classes of G.

Backtracking

Backtracking

is used to deal with most of these fundamental problems.

Idea

Use induction along a stabilizer chain:

(1) Let $G = G_1 > G_2 > \ldots > G_{l+1} = \{1\}$ defined by $G_{i+1} = Stab_{G_i}(w_i)$.

(2) Let W_i and T_i be orbit and transversal of G_{i+1} in G_i.

(3) Then each $g \in G$ can be written uniquely as $t_l \cdots t_1$ for certain $t_i \in T_i$.

(4) In particular $|G| = |T_l| \cdots |T_1|$.

Membership testing

Membership

Let $g \in Sym(n)$ and $G \leq Sym(n)$.

(1) Let $W_1 = w^{G_1}$ and determine $v = w^g$.

(2) If $v \notin W_1$, then return false.

(3) If $v \in W_1$, then there exists $t_1 \in T_1$ with $w^{t_1} = v$.

(4) Set $g_2 = gt_1^{-1}$ and iterate with W_2, etc.

(5) Yields eventually $g_{l+1} = gt_1^{-1} \cdots t_l^{-1}$.

(6) If $g_{l+1} \neq ()$, then return false.

(7) Otherwise return true, since $g = t_l \cdots t_1 \in G$.

Centralizer

Centralizer

Let $g \in Sym(n)$ and $G \leq Sym(n)$.

(1) Display the stabilizer chain as a tree.

(2) Move through the tree to determine generators for $C_G(g)$.

(3) Use some tricks as shortcuts:

- If $G_{i+1} \leq C_G(g)$ has been determined and $t_i \in C_G(g)$ is detected, then $t_i G_{i+1} \leq C_G(g)$ follows.
- If $G_{i+1} \leq C_G(g)$ has been determined and $t_i \notin C_G(g)$ is detected, then $t_i G_{i+1} \cap C_G(g) = \emptyset$ follows.

(4) It may be useful to choose a 'nice' stabilizer chain.

Examples

Examples I

```
gap> a := (1,11,2,3,4)(5,8,12,6,7);
(1,11,2,3,4)(5,8,12,6,7)
gap> b := (1,9,5,12,11,8,2,4)(6,10);
(1,9,5,12,11,8,2,4)(6,10)
gap> G := Group(a,b);
Group([ (1,11,2,3,4)(5,8,12,6,7), (1,9,5,12,11,8,2,4)(6,10) ])
gap> Size(G);
95040
gap> IsSolvable(G);
false
gap> IsSimple(G);
true
gap> IsomorphismTypeInfoFiniteSimpleGroup(G);
rec( name := "M(12)", series := "Spor" )
```

Examples I

```
gap> max := ConjugacyClassesMaximalSubgroups(G);;
gap> max := List(max, Representative);;
gap> Length(max);
11

gap> sim := Filtered(max, IsSimple);;
gap> IsomorphismTypeInfoFiniteSimpleGroup(sim[1]);
rec( name := "M(11)", series := "Spor" )
gap> IsomorphismTypeInfoFiniteSimpleGroup(sim[2]);
rec( name := "M(11)", series := "Spor" )
gap> IsomorphismTypeInfoFiniteSimpleGroup(sim[3]);
rec( name := "A(1,11) = L(2,11) ~ B(1,11) = O(3,11) ~ C(1,11)
                = S(2,11) ~ 2A(1,11) = U(2,11)",
     parameter := [ 2, 11 ],
     series := "L" )
```

Examples II

```
gap> a := (1,7,4,3,6,2,14,8,11,12,9,13);
(1,7,4,3,6,2,14,8,11,12,9,13)
gap> b := (1,9,7,15,13,6,4,12,10,3)(2,5,8,11,14);
(1,9,7,15,13,6,4,12,10,3)(2,5,8,11,14)
gap> G := Group(a,b);
Group([ (1,7,4,3,6,2,14,8,11,12,9,13), (1,9,7,15,13,6,4,12,10,3)
(2,5,8,11,14) ])
gap> Size(G);
155520
gap> IsSolvable(G);
true
gap> IsNilpotent(G);
false
gap> Collected(Factors(Size(G)));
[ [ 2, 7 ], [ 3, 5 ], [ 5, 1 ] ]
```

Examples II

```
gap> S := SylowSubgroup(G, 5);;
gap> IsNormal(G, S);
false
gap> S := SylowSubgroup(G, 3);;
gap> IsNormal(G, S);
true
gap> H := HallSubgroup(G, [2,5]);
<permutation group of size 640 with 8 generators>
```

Examples III

```
gap> G := GL(2,5);;
gap> iso := IsomorphismPermGroup(G);;
gap> H := Image(iso);
Group([ (6,11,16,21)(7,12,17,22)(8,13,18,23)(9,14,19,24)
        (10,15,20,25), (2,16, 9)(3,21,15)(4,6,17)(5,11,23)
        (7,22,10)(8,12,13)(14,18,19)(20,24,25) ])
gap> DisplayCompositionSeries(H);
G (2 gens, size 480)
 | Z(2)
S (4 gens, size 240)
 | A(5) ~ A(1,4) = L(2,4) ~ B(1,4) = O(3,4) ~ C(1,4) = S(2,4) ..
S (2 gens, size 4)
 | Z(2)
S (1 gens, size 2)
 | Z(2)
1 (0 gens, size 1)
```