Permutation Group Algorithms, Part 2

Jason B. Hill

University of Colorado

October 5, 2010

Two recent opening sentences for presentations on polynomial-time permutation group algorithms have each had five m's, one q, and one z, but this one is different in that last weeks didn't have thirteen a's, two b's, four c's, seven d's, forty-six e's, fifteen f's, five g's, thirteen h's, twenty-seven i's, one j, two k's, six l's, thirty n's, twenty o's, six p's, fifteen r's, forty s's, thirty-six t's, five u's, eleven v's, seven w's, five x's, and eight y's.

Resources:

- GAP code for Schreier-Sims functions (under this talk) at

  `http://math.jasonbhill.com/talks`

- Alexander Hulpke's "Notes on Computational Group Theory"
- Holt, et al's "Handbook of Computational Group Theory"
- Seress' "Permutation Group Algorithms"

1. Reminders

2. Stabilizer Chains and Strong Generating Sets

3. Backtrack

Reminders Of Where We Are

Last time, we reviewed the following as related to permutation groups:

- Group Actions
- Orbits and Point Stabilizers
- Various Definitions: Degree, Transitive, Primitive, Base

Reminders Of Where We Are

We developed the following algorithms in polynomial-time:

Reminders Of Where We Are

We developed the following algorithms in polynomial-time:

- “Plain Vanilla” Orbit Algorithm

Reminders Of Where We Are

We developed the following algorithms in polynomial-time:

- “Plain Vanilla” Orbit Algorithm
- Orbit Algorithm with Transversal

Reminders Of Where We Are

We developed the following algorithms in polynomial-time:

- "Plain Vanilla" Orbit Algorithm
- Orbit Algorithm with Transversal
- Orbit Algorithm with Schreier Vector

Reminders Of Where We Are

We developed the following algorithms in polynomial-time:

- "Plain Vanilla" Orbit Algorithm
- Orbit Algorithm with Transversal
- Orbit Algorithm with Schreier Vector
- Orbit Stabilizer Algorithm (as a consequence of Schreier's Theorem)

Reminders Of Where We Are

We developed the following algorithms in polynomial-time:

- "Plain Vanilla" Orbit Algorithm
- Orbit Algorithm with Transversal
- Orbit Algorithm with Schreier Vector
- Orbit Stabilizer Algorithm (as a consequence of Schreier's Theorem)
- Normal Closure (We didn't explain this entirely. In essence, we need some closure here. We'll assume for right now that this is possible in P.)

Reminders Of Where We Are

This allows us to calculate the following in polynomial-time:

Reminders Of Where We Are

This allows us to calculate the following in polynomial-time:

- Orbits and Transitivity

Reminders Of Where We Are

This allows us to calculate the following in polynomial-time:

- Orbits and Transitivity
- Transversals and Generators for Point Stabilizers

Reminders Of Where We Are

This allows us to calculate the following in polynomial-time:

- Orbits and Transitivity
- Transversals and Generators for Point Stabilizers
- Normal Closure for Subgroups

Reminders Of Where We Are

This allows us to calculate the following in polynomial-time:

- Orbits and Transitivity
- Transversals and Generators for Point Stabilizers
- Normal Closure for Subgroups
- For $N \leq G$ test if $N \lhd G$

Reminders Of Where We Are

This allows us to calculate the following in polynomial-time:

- Orbits and Transitivity
- Transversals and Generators for Point Stabilizers
- Normal Closure for Subgroups
- For $N \leq G$ test if $N \lhd G$
- Commutator Subgroups $G' = \langle a^{-1}b^{-1}ab \mid a, b \in \underline{g} \rangle_G$

Reminders Of Where We Are

This allows us to calculate the following in polynomial-time:

- Orbits and Transitivity
- Transversals and Generators for Point Stabilizers
- Normal Closure for Subgroups
- For $N \leq G$ test if $N \lhd G$
- Commutator Subgroups $G' = \langle a^{-1}b^{-1}ab \mid a, b \in \underline{g} \rangle_G$
- Derived and Lower Central Series

Reminders Of Where We Are

This allows us to calculate the following in polynomial-time:

- Orbits and Transitivity
- Transversals and Generators for Point Stabilizers
- Normal Closure for Subgroups
- For $N \leq G$ test if $N \lhd G$
- Commutator Subgroups $G' = \langle a^{-1}b^{-1}ab \mid a, b \in \underline{g} \rangle_G$
- Derived and Lower Central Series
- Test if G is Solvable or Nilpotent

Reminders Of Where We Are

Goals For Today:

- Stabilizer Chains
- Schreier-Sims Algorithm
- Introduction to Backtrack

Stabilizer Chains

The fundamental ideas behind the resulting algorithm were developed by Sims (1970) and uses Schreier's Theorem.

Stabilizer Chains

The fundamental ideas behind the resulting algorithm were developed by Sims (1970) and uses Schreier's Theorem.

- Let $G = \langle \underline{g} \rangle$ be a permutation group acting on Ω with $|\Omega| = n$.

Stabilizer Chains

The fundamental ideas behind the resulting algorithm were developed by Sims (1970) and uses Schreier's Theorem.

- Let $G = \langle \underline{g} \rangle$ be a permutation group acting on Ω with $|\Omega| = n$.
- Let $B = [\beta_1, \ldots, \beta_k] \subseteq \Omega$ (typically a base)

Stabilizer Chains

The fundamental ideas behind the resulting algorithm were developed by Sims (1970) and uses Schreier's Theorem.

- Let $G = \langle \underline{g} \rangle$ be a permutation group acting on Ω with $|\Omega| = n$.
- Let $B = [\beta_1, \ldots, \beta_k] \subseteq \Omega$ (typically a base)

Stabilizer Chain

A stabilizer chain for G with respect to B is the chain of subgroups defined by $G^{(0)} = G$, $G^{(1)} = G_{\beta_1} = \mathrm{Stab}_G(\beta_1)$, and

$$G^{(i)} = G_{\beta_i}^{G^{(i-1)}} = \mathrm{Stab}_{G^{(i-1)}}(\beta_i) = \mathrm{Stab}_G(\beta_1, \ldots, \beta_i)$$

where the last group may be viewed as a tuple (not set) stabilizer.

Stabilizer Chains

Example: Let $G = S_5$ act on $\Omega = [1 \dots 5]$ with $B = [1 \dots 4]$.

$$\begin{array}{ll}
 & G^{(0)} = S_5 \\
5 & \\
 & G^{(1)} = G_1^{(0)} = S_5|_{[2\dots5]} \simeq S_4 \text{ on } [2 \dots 5] \\
4 & \\
 & G^{(2)} = G_2^{(1)} = S_5|_{[3\dots5]} \simeq S_3 \text{ on } [3 \dots 5] \\
3 & \\
 & G^{(3)} = G_3^{(2)} = S_5|_{[4,5]} \simeq S_2 \text{ on } [4, 5] \simeq Z_2 \\
2 & \\
 & G^{(4)} = G_4^{(3)} = S_5|_{[5]} \simeq \langle\,\rangle
\end{array}$$

Stabilizer Chains

Some Notes:

Stabilizer Chains

Some Notes:

- By definition, when B is a base of length k, we have $G^{(k)} = \langle\, \rangle$.

Stabilizer Chains

Some Notes:

- By definition, when B is a base of length k, we have $G^{(k)} = \langle\,\rangle$.
- In this situation, $|G| = \prod_{i=1}^{k} \left[G^{(i-1)} : G^{(i)} \right] = \prod_{i=1}^{k} |T^{(i)}| = \prod_{i=1}^{k} |\Delta^{(i)}|$

Stabilizer Chains

Some Notes:

- By definition, when B is a base of length k, we have $G^{(k)} = \langle\,\rangle$.
- In this situation, $|G| = \prod_{i=1}^{k} \left[G^{(i-1)} : G^{(i)} \right] = \prod_{i=1}^{k} |T^{(i)}| = \prod_{i=1}^{k} |\Delta^{(i)}|$
- Schreier's Theorem yields (Schreier) generators for each $G^{(i)}$ ($i \geq 1$).

Stabilizer Chains

Main Idea: Sifting Let $g \in G$, $B = [\beta_1 \dots \beta_k]$ be a base.

Stabilizer Chains

Main Idea: Sifting Let $g \in G$, $B = [\beta_1 \dots \beta_k]$ be a base.

- g is in a coset of $G^{(1)}$, so $g = a_1 \overline{b_1}$ for $a_1 \in G^{(1)}$, $\overline{b_1} \in {G^{(1)}}^{\backslash G}$

Stabilizer Chains

Main Idea: Sifting Let $g \in G$, $B = [\beta_1 \ldots \beta_k]$ be a base.

- g is in a coset of $G^{(1)}$, so $g = a_1\overline{b_1}$ for $a_1 \in G^{(1)}$, $\overline{b_1} \in G^{(1)^{\backslash G}}$
- Then $a_1 = a_2\overline{b_2}$ for $a_2 \in G^{(2)}$, $\overline{b_2} \in G^{(2)^{\backslash G^{(1)}}}$. (So $g = a_2\overline{b_2 b_1}$.)

Stabilizer Chains

Main Idea: Sifting Let $g \in G$, $B = [\beta_1 \dots \beta_k]$ be a base.

- g is in a coset of $G^{(1)}$, so $g = a_1\overline{b_1}$ for $a_1 \in G^{(1)}$, $\overline{b_1} \in G^{(1)^{\backslash G}}$
- Then $a_1 = a_2\overline{b_2}$ for $a_2 \in G^{(2)}$, $\overline{b_2} \in G^{(2)^{\backslash G^{(1)}}}$. (So $g = a_2\overline{b_2 b_1}$.)
- Inductively, each $g \in G$ has a (unique) decomposition

$$g = \overline{b_k b_{k-1} \cdots b_2 b_1} \quad \text{with} \quad \overline{b_i} \in G^{(i)^{\backslash G^{(i-1)}}}.$$

Stabilizer Chains

Main Idea: Sifting Let $g \in G$, $B = [\beta_1 \ldots \beta_k]$ be a base.

- g is in a coset of $G^{(1)}$, so $g = a_1 \overline{b_1}$ for $a_1 \in G^{(1)}$, $\overline{b_1} \in {G^{(1)}}^{\backslash G}$
- Then $a_1 = a_2 \overline{b_2}$ for $a_2 \in G^{(2)}$, $\overline{b_2} \in {G^{(2)}}^{\backslash G^{(1)}}$. (So $g = a_2 \overline{b_2 b_1}$.)
- Inductively, each $g \in G$ has a (unique) decomposition

$$g = \overline{b_k b_{k-1} \cdots b_2 b_1} \quad \text{with} \quad \overline{b_i} \in {G^{(i)}}^{\backslash G^{(i-1)}}.$$

- The transversal elements $\overline{b_i}$ (found via a Schreier vector) and corresponding orbit points, plus generators for the stabilizers at each level in the chain are given by the Orbit-Stabilizer Algorithm discussed last week.

Stabilizer Chains

Since each element $g \in G$ can be factored by a list of coset representatives having length at most the size of the base, this process does allow us to discuss every element in a permutation group while only maintaining relatively few group elements in memory.

Stabilizer Chains

Since each element $g \in G$ can be factored by a list of coset representatives having length at most the size of the base, this process does allow us to discuss every element in a permutation group while only maintaining relatively few group elements in memory.

Several Considerations

Stabilizer Chains

Since each element $g \in G$ can be factored by a list of coset representatives having length at most the size of the base, this process does allow us to discuss every element in a permutation group while only maintaining relatively few group elements in memory.

Several Considerations

- A long stabilizer chain results in storing many levels of Schreier generators. Can we simplify this by somehow storing all required generators at a single level?

Stabilizer Chains

Since each element $g \in G$ can be factored by a list of coset representatives having length at most the size of the base, this process does allow us to discuss every element in a permutation group while only maintaining relatively few group elements in memory.

Several Considerations

- A long stabilizer chain results in storing many levels of Schreier generators. Can we simplify this by somehow storing all required generators at a single level? (Ans: Yes)

Stabilizer Chains

Since each element $g \in G$ can be factored by a list of coset representatives having length at most the size of the base, this process does allow us to discuss every element in a permutation group while only maintaining relatively few group elements in memory.

Several Considerations

- A long stabilizer chain results in storing many levels of Schreier generators. Can we simplify this by somehow storing all required generators at a single level? (Ans: Yes)
- What happens if we do not have a base precomputed? Can we modify the creation of the stabilizer chain to produce a base?

Stabilizer Chains

Since each element $g \in G$ can be factored by a list of coset representatives having length at most the size of the base, this process does allow us to discuss every element in a permutation group while only maintaining relatively few group elements in memory.

Several Considerations

- A long stabilizer chain results in storing many levels of Schreier generators. Can we simplify this by somehow storing all required generators at a single level? (Ans: Yes)
- What happens if we do not have a base precomputed? Can we modify the creation of the stabilizer chain to produce a base? (Ans: Yes)

Schreier-Sims Algorithm

Schreier-Sims Algorithm

Strong Generating System (SGS)

A generating set $\underline{g}$ is called a strong generating set for G relative to a base B, if $\langle \underline{g} \cap G^{(i)} \rangle = G^{(i)}$ for $0 \leq i \leq |B| - 1$. Hence, a stabilizer chain computed from an SGS will have transversal products as words in $\underline{g}$.

Schreier-Sims Algorithm

Strong Generating System (SGS)

A generating set $\underline{g}$ is called a strong generating set for G relative to a base B, if $\langle \underline{g} \cap G^{(i)} \rangle = G^{(i)}$ for $0 \leq i \leq |B| - 1$. Hence, a stabilizer chain computed from an SGS will have transversal products as words in $\underline{g}$.

BSGS

A BSGS for a group G consists of a base B together with a strong generating set (SGS) relative to B.

Schreier-Sims Algorithm

Strong Generating System (SGS)

A generating set $\underline{g}$ is called a <u>strong generating set</u> for G relative to a base B, if $\langle \underline{g} \cap G^{(i)} \rangle = G^{(i)}$ for $0 \leq i \leq |B| - 1$. Hence, a stabilizer chain computed from an SGS will have transversal products as words in $\underline{g}$.

BSGS

A <u>BSGS</u> for a group G consists of a base B together with a strong generating set (SGS) relative to B.

Schreier-Sims Algorithm (Sims, 1970)

- **Input:** an arbitrary generating set $\underline{g}$, optional partial/full base.
- **Output:** BSGS

Schreier-Sims Algorithm

Details: See the handout.

Schreier-Sims Algorithm

Details: See the handout.

Main Ideas:

Schreier-Sims Algorithm

Details: See the handout.

Main Ideas:

- Construct a stabilizer chain. Instead of adding Schreier generators at each level, determine if those generators can be sifted with the existing chain's Schreier vectors. If not, add them to the SGS.

Schreier-Sims Algorithm

Details: See the handout.

Main Ideas:

- Construct a stabilizer chain. Instead of adding Schreier generators at each level, determine if those generators can be sifted with the existing chain's Schreier vectors. If not, add them to the SGS.
- If the base is not long enough, the group will not factor entirely from the given chain and a new base element is needed.

Schreier-Sims Algorithm

Details: See the handout.

Main Ideas:

- Construct a stabilizer chain. Instead of adding Schreier generators at each level, determine if those generators can be sifted with the existing chain's Schreier vectors. If not, add them to the SGS.
- If the base is not long enough, the group will not factor entirely from the given chain and a new base element is needed.
- Whenever the calculated BSGS changes, initialize the stabilizer chain calculation again.

Schreier-Sims Algorithm

Realistic Example: See handout.

Schreier-Sims Algorithm

Realistic Example: See handout.

Simple Example: A BSGS for S_3 with no input base.

Schreier-Sims Algorithm

Realistic Example: See handout.

Simple Example: A BSGS for S_3 with no input base.

```
gap> jbhSchreierSims([(1,2,3),(1,2)],[]);
[ [ (), (2,3), (1,2), (1,2,3) ], [ 1, 2 ] ]
```

Schreier-Sims Algorithm

Realistic Example: See handout.

Simple Example: A BSGS for S_3 with no input base.

```
gap> jbhSchreierSims([(1,2,3),(1,2)],[]);
[ [ (), (2,3), (1,2), (1,2,3) ], [ 1, 2 ] ]
```

This corresponds to the stabilizer chain:

Group	Generators	Orbit	Schreier Vector
$G^{(0)}$	$[(), (2,3), (1,2), (1,2,3)]$	$[1,2,3]$	$[(), 3, 2]$
$G^{(1)}$	$[(), (2,3)]$	$[2,3]$	$[(), 2]$
$G^{(2)}$	$[()]$	$[]$	$[()]$

Schreier-Sims Algorithm

- Furst, Hopcroft and Luks (1980) showed that the deterministic Schreier-Sims has running time $O(n^6 + kn^2)$.

Schreier-Sims Algorithm

- Furst, Hopcroft and Luks (1980) showed that the deterministic Schreier-Sims has running time $O(n^6 + kn^2)$.
- D. Knuth and Jerrum each improved the timing to $O(n^5 + kn^2)$ (1981,1982 resp.)

Schreier-Sims Algorithm

- Furst, Hopcroft and Luks (1980) showed that the deterministic Schreier-Sims has running time $O(n^6 + kn^2)$.
- D. Knuth and Jerrum each improved the timing to $O(n^5 + kn^2)$ (1981,1982 resp.)
- Sims (1990) showed that a nearly-linear-time deterministic algorithm was possible for solvable groups.

Schreier-Sims Algorithm

- Furst, Hopcroft and Luks (1980) showed that the deterministic Schreier-Sims has running time $O(n^6 + kn^2)$.
- D. Knuth and Jerrum each improved the timing to $O(n^5 + kn^2)$ (1981,1982 resp.)
- Sims (1990) showed that a nearly-linear-time deterministic algorithm was possible for solvable groups.

- We shall briefly discuss some improved versions of Schreier-Sims, and then consider consequences in the remainder of this talk.

Giant Groups

- $\mathrm{Alt}(\Omega)$ and $\mathrm{Sym}(\Omega)$ are "large base" groups, and thus have lengthy stabilizer chains. A BSGS calculation via a deterministic and non-optimized Schreier-Sims can be costly.

Giant Groups

- $\mathrm{Alt}(\Omega)$ and $\mathrm{Sym}(\Omega)$ are "large base" groups, and thus have lengthy stabilizer chains. A BSGS calculation via a deterministic and non-optimized Schreier-Sims can be costly.
- The jbhSchreierSims GAP function (was never meant to be efficient) took approximately 50 hours to completely factor $\mathrm{Alt}([1 \ldots 450])$.

Giant Groups

- $\mathrm{Alt}(\Omega)$ and $\mathrm{Sym}(\Omega)$ are "large base" groups, and thus have lengthy stabilizer chains. A BSGS calculation via a deterministic and non-optimized Schreier-Sims can be costly.
- The jbhSchreierSims GAP function (was never meant to be efficient) took approximately 50 hours to completely factor $\mathrm{Alt}([1 \ldots 450])$.

Giant Group Theorem

Let $G \leq \mathrm{Sym}(\Omega)$ and $|\Omega| = n$. If there exists $g \in G$ with a cycle of prime length p satisfying $\frac{n}{2} < p < n - 2$, then $\mathrm{Alt}(\Omega) \leq G$.

Giant Groups

- $\mathrm{Alt}(\Omega)$ and $\mathrm{Sym}(\Omega)$ are "large base" groups, and thus have lengthy stabilizer chains. A BSGS calculation via a deterministic and non-optimized Schreier-Sims can be costly.
- The jbhSchreierSims GAP function (was never meant to be efficient) took approximately 50 hours to completely factor $\mathrm{Alt}([1 \ldots 450])$.

Giant Group Theorem

Let $G \leq \mathrm{Sym}(\Omega)$ and $|\Omega| = n$. If there exists $g \in G$ with a cycle of prime length p satisfying $\frac{n}{2} < p < n-2$, then $\mathrm{Alt}(\Omega) \leq G$.

- $\mathrm{Alt}(\Omega)$ and $\mathrm{Sym}(\Omega)$ have known BSGS structures.
- If a giant group is found, then consider $\mathrm{sgn}(g)$ for $g \in \underline{g}$.

Random Schreier-Sims

Random Schreier-Sims

Random Schreier-Sims

Random Schreier-Sims

- Let $(B, \underline{g})$ be an input base and generating set. If this is not a BSGS, then the sifting process in Schreier-Sims will return a non-identity element with probability at least $1/2$.

Random Schreier-Sims

Random Schreier-Sims

- Let $(B, \underline{g})$ be an input base and generating set. If this is not a BSGS, then the sifting process in Schreier-Sims will return a non-identity element with probability at least $1/2$.
- We may then add the sifted element to the $\underline{g}$ and recalculate a base.

Random Schreier-Sims

Random Schreier-Sims

- Let $(B, \underline{g})$ be an input base and generating set. If this is not a BSGS, then the sifting process in Schreier-Sims will return a non-identity element with probability at least $1/2$.
- We may then add the sifted element to the $\underline{g}$ and recalculate a base.
- Picking enough random elements (≈ 10 with some uniform distribution) has a high probability of providing a SGS.

Schreier-Sims Algorithm

Consequences: Given a BSGS for a permutation group G acting on Ω.

Schreier-Sims Algorithm

Consequences: Given a BSGS for a permutation group G acting on Ω.

- $|G|$ is, by Lagrange's Theorem and the Orbit-Stabilizer Theorem, the product of orbit lengths (also size of transversals / Schreier vectors) in the stabilizer chain.

Schreier-Sims Algorithm

Consequences: Given a BSGS for a permutation group G acting on Ω.

- $|G|$ is, by Lagrange's Theorem and the Orbit-Stabilizer Theorem, the product of orbit lengths (also size of transversals / Schreier vectors) in the stabilizer chain.
- Thus, we may enumerate all elements of G by a process known as "backtrack," where we consider all possible products of transversals in some organized fashion.

Schreier-Sims Algorithm

Consequences: Given a BSGS for a permutation group G acting on Ω.

- $|G|$ is, by Lagrange's Theorem and the Orbit-Stabilizer Theorem, the product of orbit lengths (also size of transversals / Schreier vectors) in the stabilizer chain.
- Thus, we may enumerate all elements of G by a process known as "backtrack," where we consider all possible products of transversals in some organized fashion.
- Given a permutation g we may test $g \in G$ by attempting to factor via coset representatives.

Schreier-Sims Algorithm

Consequences: Given a BSGS for a permutation group G acting on Ω.

- $|G|$ is, by Lagrange's Theorem and the Orbit-Stabilizer Theorem, the product of orbit lengths (also size of transversals / Schreier vectors) in the stabilizer chain.
- Thus, we may enumerate all elements of G by a process known as "backtrack," where we consider all possible products of transversals in some organized fashion.
- Given a permutation g we may test $g \in G$ by attempting to factor via coset representatives.
- The same process allows us to test membership in subgroups (as we need to do in the normal closure algorithm).

Schreier-Sims Algorithm

Consequences: Given a BSGS for a permutation group G acting on Ω.

- $|G|$ is, by Lagrange's Theorem and the Orbit-Stabilizer Theorem, the product of orbit lengths (also size of transversals / Schreier vectors) in the stabilizer chain.
- Thus, we may enumerate all elements of G by a process known as "backtrack," where we consider all possible products of transversals in some organized fashion.
- Given a permutation g we may test $g \in G$ by attempting to factor via coset representatives.
- The same process allows us to test membership in subgroups (as we need to do in the normal closure algorithm).
- Obtain random elements with guaranteed equal distribution.

Example of Testing Group Membership

Group Membership in Dihedral Group of Degree 6

```
gap> D6_BSGS:=jbhSchreierSims([(1,2,3,4,5,6),(2,6)(3,5)],[]);;
```

Example of Testing Group Membership

Group Membership in Dihedral Group of Degree 6

```
gap> D6_BSGS:=jbhSchreierSims([(1,2,3,4,5,6),(2,6)(3,5)],[]);;
gap> D6_Chain:=jbhStabilizerChainStrong(D6_BSGS[1],D6_BSGS[2]);;
```

Example of Testing Group Membership

Group Membership in Dihedral Group of Degree 6

```
gap> D6_BSGS:=jbhSchreierSims([(1,2,3,4,5,6),(2,6)(3,5)],[]);;
gap> D6_Chain:=jbhStabilizerChainStrong(D6_BSGS[1],D6_BSGS[2]);;
gap> jbhFactorOrFailFP((1,3,5)(2,4,6),D6_BSGS[2],D6_Chain);
```

Example of Testing Group Membership

Group Membership in Dihedral Group of Degree 6

```
gap> D6_BSGS:=jbhSchreierSims([(1,2,3,4,5,6),(2,6)(3,5)],[]);;
gap> D6_Chain:=jbhStabilizerChainStrong(D6_BSGS[1],D6_BSGS[2]);;
gap> jbhFactorOrFailFP((1,3,5)(2,4,6),D6_BSGS[2],D6_Chain);
Factorization:  (1,3,5)(2,4,6)=(1,5,3)(2,6,4)
[ (), 3 ]  (Factorization Complete)
```

Example of Testing Group Membership

Group Membership in Dihedral Group of Degree 6

```
gap> D6_BSGS:=jbhSchreierSims([(1,2,3,4,5,6),(2,6)(3,5)],[]);;
gap> D6_Chain:=jbhStabilizerChainStrong(D6_BSGS[1],D6_BSGS[2]);;
gap> jbhFactorOrFailFP((1,3,5)(2,4,6),D6_BSGS[2],D6_Chain);
Factorization:  (1,3,5)(2,4,6)=(1,5,3)(2,6,4)
[ (), 3 ]  (Factorization Complete)
gap> jbhFactorOrFailFP((1,2,3,4),D6_BSGS[2],D6_Chain);
```

Example of Testing Group Membership

Group Membership in Dihedral Group of Degree 6

```
gap> D6_BSGS:=jbhSchreierSims([(1,2,3,4,5,6),(2,6)(3,5)],[]);;
gap> D6_Chain:=jbhStabilizerChainStrong(D6_BSGS[1],D6_BSGS[2]);;
gap> jbhFactorOrFailFP((1,3,5)(2,4,6),D6_BSGS[2],D6_Chain);
Factorization:  (1,3,5)(2,4,6)=(1,5,3)(2,6,4)
[ (), 3 ]  (Factorization Complete)
gap> jbhFactorOrFailFP((1,2,3,4),D6_BSGS[2],D6_Chain);
Factorization:  (1,2,3,4)=(1,6,5,4,3,2)
[ (4,6,5), 3 ]  (Factorization Not Complete)
Fail
gap>
```

Backtrack

Backtrack

- Searching through an entire group results in a runtime of $O(|G|)$.

Backtrack

- Searching through an entire group results in a runtime of $O(|G|)$.
- In general, this is not a polynomial in the degree of the group.

Backtrack

- Searching through an entire group results in a runtime of $O(|G|)$.
- In general, this is not a polynomial in the degree of the group.
- We will now organize such a search, known as **Backtrack**.

Backtrack

- Searching through an entire group results in a runtime of $O(|G|)$.
- In general, this is not a polynomial in the degree of the group.
- We will now organize such a search, known as **Backtrack**.

- Assume we have a BSGS $(B, \underline{g})$ for G.

Backtrack

- Searching through an entire group results in a runtime of $O(|G|)$.
- In general, this is not a polynomial in the degree of the group.
- We will now organize such a search, known as **Backtrack**.

- Assume we have a BSGS $(B, \underline{g})$ for G.
- Recall that $B^g = [\beta_1^g, \ldots, \beta_k^g]$ uniquely determines each $g \in G$.

Backtrack

- Searching through an entire group results in a runtime of $O(|G|)$.
- In general, this is not a polynomial in the degree of the group.
- We will now organize such a search, known as **Backtrack**.

- Assume we have a BSGS $(B, \underline{g})$ for G.
- Recall that $B^g = [\beta_1^g, \ldots, \beta_k^g]$ uniquely determines each $g \in G$.
- First, we create an order relative to B^g.

Backtrack

Fix a Base $B = [\beta_1, \ldots, \beta_k]$ for G.

Backtrack

Fix a Base $B = [\beta_1, \ldots, \beta_k]$ for G.

The order $\prec$ on Ω

Define an order $\prec$ on $\Omega \subset \mathbb{Z}_{>0}$ such that

1. For $\beta_i, \beta_j \in B$, $\beta_i \prec \beta_j$ if $i < j$.
2. For $\alpha, \gamma \in \Omega \backslash B$, $\alpha \prec \gamma$ if $\alpha < \gamma$.
3. If $\beta \in B$ and $\alpha \in \Omega \backslash B$ then $\beta \prec \alpha$.

Backtrack

Fix a Base $B = [\beta_1, \ldots, \beta_k]$ for G.

The order $\prec$ on Ω

Define an order $\prec$ on $\Omega \subset \mathbb{Z}_{>0}$ such that

1. For $\beta_i, \beta_j \in B$, $\beta_i \prec \beta_j$ if $i < j$.
2. For $\alpha, \gamma \in \Omega \backslash B$, $\alpha \prec \gamma$ if $\alpha < \gamma$.
3. If $\beta \in B$ and $\alpha \in \Omega \backslash B$ then $\beta \prec \alpha$.

The order $\prec$ on G

Define the order $\prec$ on G such that $g \prec h$ if and only if for some $\ell \leq k$ we have $\beta_i^g = \beta_i^h$ for $i < \ell$ and $\beta_\ell^g \prec \beta_\ell^h$.

Backtrack

Fix a Base $B = [\beta_1, \ldots, \beta_k]$ for G.

The order $\prec$ on Ω

Define an order $\prec$ on $\Omega \subset \mathbb{Z}_{>0}$ such that

1. For $\beta_i, \beta_j \in B$, $\beta_i \prec \beta_j$ if $i < j$.
2. For $\alpha, \gamma \in \Omega \backslash B$, $\alpha \prec \gamma$ if $\alpha < \gamma$.
3. If $\beta \in B$ and $\alpha \in \Omega \backslash B$ then $\beta \prec \alpha$.

The order $\prec$ on G

Define the order $\prec$ on G such that $g \prec h$ if and only if for some $\ell \leq k$ we have $\beta_i^g = \beta_i^h$ for $i < \ell$ and $\beta_\ell^g \prec \beta_\ell^h$.

- By definition of base, the identity element will always be least w.r.t $\prec$.

Backtrack

Example:

- Consider the group $D_8 \times Z_2 \simeq$ `Group((1,2,3,4),(2,4),(5,6))`.

Backtrack

Example:

- Consider the group $D_8 \times Z_2 \simeq$ `Group((1,2,3,4),(2,4),(5,6))`.
- The Schreier-Sims algorithm finds:
  - Strong Generating Set: `[(), (5,6), (2,4), (1,2,3,4) ]`
  - Base: `[ 1, 2, 5 ]`

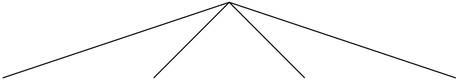
Backtrack

Example:

- Consider the group $D_8 \times Z_2 \simeq$ `Group((1,2,3,4),(2,4),(5,6))`.
- The Schreier-Sims algorithm finds:
  - Strong Generating Set: `[(), (5,6), (2,4), (1,2,3,4) ]`
  - Base: `[ 1, 2, 5 ]`
- A few base images (by order of $\prec$):
  - `[ 1, 2, 5, 3, 4, 6 ]=()`
  - `[ 1, 2, 6, 3, 4, 5 ]=(5,6)`
  - `[ 1, 4, 5, 3, 2, 6 ]=(2,4)`
  - `[ 1, 4, 6, 3, 2, 5 ]=(2,4)(5,6)`

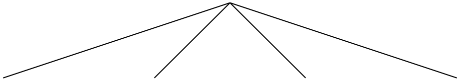
Searching Through Group Elements

Main Idea: Create a tree, where nodes represent transversal products and branches represent stabilizer cosets.



Searching Through Group Elements

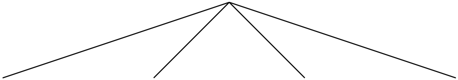
Main Idea: Create a tree, where nodes represent transversal products and branches represent stabilizer cosets.



- Essentially, we consider all elements of G in $\prec$-order.

Searching Through Group Elements

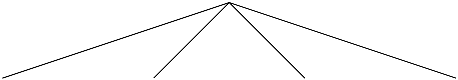
Main Idea: Create a tree, where nodes represent transversal products and branches represent stabilizer cosets.



- Essentially, we consider all elements of G in $\prec$-order.
- Start with cosets of $G^{(1)}$ as branches.

Searching Through Group Elements

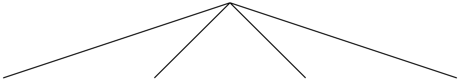
Main Idea: Create a tree, where nodes represent transversal products and branches represent stabilizer cosets.



- Essentially, we consider all elements of G in $\prec$-order.
- Start with cosets of $G^{(1)}$ as branches.
- The identity coset is farthest to the left, fixing the first base element.

Searching Through Group Elements

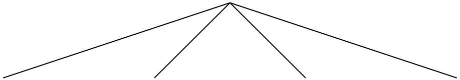
Main Idea: Create a tree, where nodes represent transversal products and branches represent stabilizer cosets.



- Essentially, we consider all elements of G in $\prec$-order.
- Start with cosets of $G^{(1)}$ as branches.
- The identity coset is farthest to the left, fixing the first base element.
- Order the other cosets from left to right based on the base image.

Searching Through Group Elements

Main Idea: Create a tree, where nodes represent transversal products and branches represent stabilizer cosets.



- Essentially, we consider all elements of G in $\prec$-order.
- Start with cosets of $G^{(1)}$ as branches.
- The identity coset is farthest to the left, fixing the first base element.
- Order the other cosets from left to right based on the base image.
- The cosets of $G^{(2)}$ then give branches on the next level.

Backtrack: Tree Example

Let $G = S_3$ with $SGS = [(), (2,3), (1,2), (1,2,3)]$ with base $[1,2]$.

Backtrack: Tree Example

Let $G = S_3$ with $SGS = [(), (2,3), (1,2), (1,2,3)]$ with base $[1,2]$.

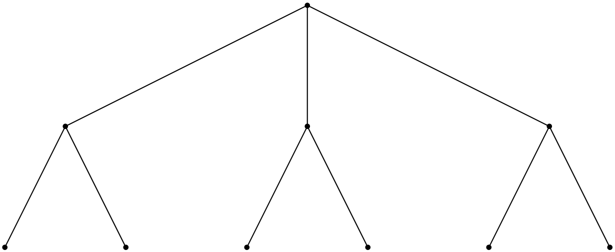
The first transversal in the stabilizer chain is $T^{(1)} = [(), (1,2), (1,2)(2,3)]$

Backtrack: Tree Example

Let $G = S_3$ with $SGS = [(), (2,3), (1,2), (1,2,3)]$ with base $[1,2]$.

The first transversal in the stabilizer chain is $T^{(1)} = [(), (1,2), (1,2)(2,3)]$

The second transversal is $T^{(2)} = [(), (2,3)]$.

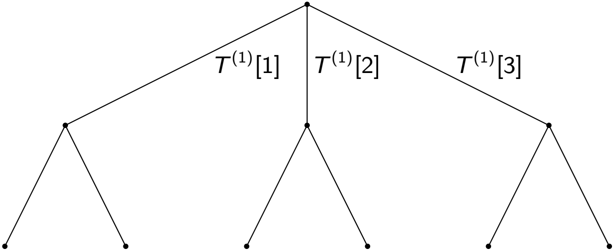


Backtrack: Tree Example

Let $G = S_3$ with $SGS = [(), (2,3), (1,2), (1,2,3)]$ with base $[1,2]$.

The first transversal in the stabilizer chain is $T^{(1)} = [(), (1,2), (1,2)(2,3)]$

The second transversal is $T^{(2)} = [(), (2,3)]$.

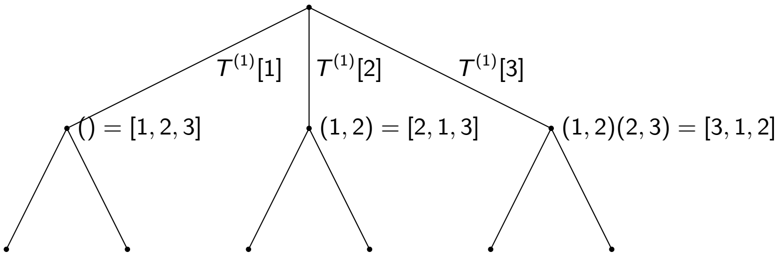


Backtrack: Tree Example

Let $G = S_3$ with $SGS = [(), (2,3), (1,2), (1,2,3)]$ with base $[1,2]$.

The first transversal in the stabilizer chain is $T^{(1)} = [(), (1,2), (1,2)(2,3)]$

The second transversal is $T^{(2)} = [(), (2,3)]$.

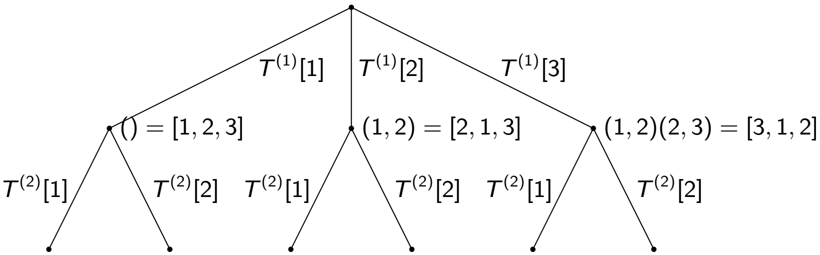


Backtrack: Tree Example

Let $G = S_3$ with $SGS = [(), (2,3), (1,2), (1,2,3)]$ with base $[1,2]$.

The first transversal in the stabilizer chain is $T^{(1)} = [(), (1,2), (1,2)(2,3)]$

The second transversal is $T^{(2)} = [(), (2,3)]$.

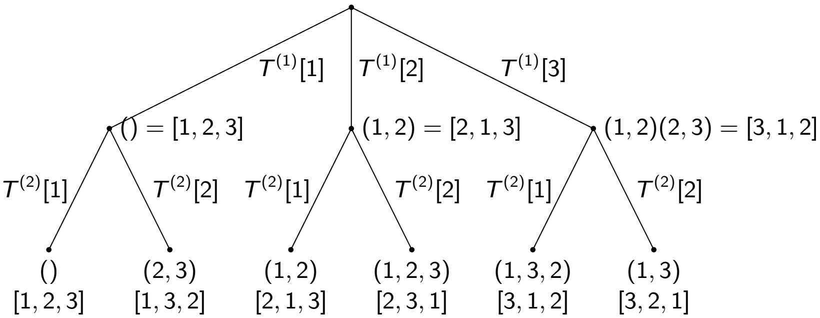


Backtrack: Tree Example

Let $G = S_3$ with $SGS = [(), (2,3), (1,2), (1,2,3)]$ with base $[1,2]$.

The first transversal in the stabilizer chain is $T^{(1)} = [(), (1,2), (1,2)(2,3)]$

The second transversal is $T^{(2)} = [(), (2,3)]$.



Backtrack

Some Notes:

Backtrack

Some Notes:

- We may need to order vertices under a node to conform to the order $\prec$ since Schreier vectors are not generated with this order in mind.

Backtrack

Some Notes:

- We may need to order vertices under a node to conform to the order $\prec$ since Schreier vectors are not generated with this order in mind.
- The result is a tree with final level in $\prec$-order.

Backtrack

Some Notes:

- We may need to order vertices under a node to conform to the order $\prec$ since Schreier vectors are not generated with this order in mind.
- The result is a tree with final level in $\prec$-order.
- At each nontrivial level ℓ, base images are partially defined. That is, at level ℓ the image β_i^g is defined for $1 \leq i \leq \ell$.

Backtrack

Some Notes:

- We may need to order vertices under a node to conform to the order $\prec$ since Schreier vectors are not generated with this order in mind.
- The result is a tree with final level in $\prec$-order.
- At each nontrivial level ℓ, base images are partially defined. That is, at level ℓ the image β_i^g is defined for $1 \leq i \leq \ell$.
- Thus, if we could *somehow* limit our search for elements based on the initial base image by some $\ell < k$, we would reduce the search.

Backtrack

Some Notes:

- We may need to order vertices under a node to conform to the order $\prec$ since Schreier vectors are not generated with this order in mind.
- The result is a tree with final level in $\prec$-order.
- At each nontrivial level ℓ, base images are partially defined. That is, at level ℓ the image β_i^g is defined for $1 \leq i \leq \ell$.
- Thus, if we could *somehow* limit our search for elements based on the initial base image by some $\ell < k$, we would reduce the search.
- This process is known as "pruning the search tree" as it removes branches below nodes.

Backtrack

Some Notes:

- We may need to order vertices under a node to conform to the order $\prec$ since Schreier vectors are not generated with this order in mind.
- The result is a tree with final level in $\prec$-order.
- At each nontrivial level ℓ, base images are partially defined. That is, at level ℓ the image β_i^g is defined for $1 \le i \le \ell$.
- Thus, if we could *somehow* limit our search for elements based on the initial base image by some $\ell < k$, we would reduce the search.
- This process is known as "pruning the search tree" as it removes branches below nodes.

Next week, we will consider this process in more depth and look at specific problems that reduce to this procedure. Also, we'll consider improvements using group structure.

Thank You