

An Introduction to Large Permutation Groups

Jeffrey Clark

Department of Mathematics, Elon College

Elon College, NC 27244

clarkj@elon.edu

January 22, 2000

Abstract

Large permutation groups can be factored into subgroup chains; it is possible to describe briefly the properties of groups too large to list explicitly.

1 Introduction

When I teach groups in my Abstract Algebra class, we focus almost entirely upon ones that can be described by a multiplication table. Most of the good examples that I cite in class (symmetry groups for polyhedra, the permutation group for Rubik's cube) are too large to be so described, even when they are finite.

How can we know that 43,252,003,274,489,856,000 is the size of the permutation group for Rubik's cube? The number is far too large to find by listing all the group elements, even by computer. This paper will describe a method for finding the size of a permutation group that will also provide a method for testing whether or not a given permutation is contained within the group.

In what follows, the work on orbits and stabilizers can be found in elementary texts on group theory such as [4] and [5], while the work on Schreier's Theorem is less well-known and can be found in [1], [2], [3], and [6].

2 Strategy

We will break down our large group, one step at a time, into a chain of nested subgroups.

Let G be a permutation group on $\{1, \dots, n\}$, i.e., a subgroup of S_n , that is given by a list of generators $\{p_1, \dots, p_m\}$. We want to be able to take the list of generators to find the size of G , and along the way, develop a test for membership in G .

At each stage, we will pick a point x in the set $\{1, \dots, n\}$ being permuted, and look at

1. its orbit, which can be found using the generators of the group, and
2. its stabilizer.

2.1 Orbits and Stabilizers

For any point x between 1 and n and any $g \in G$, we define x^g to be the action of g on x . We then have for each x

1. $x^G = \{x^g \mid g \in G\}$ is the orbit of x , and
2. $G_x = \{g \in G \mid x^g = x\}$ is the stabilizer of x .

Let $g, h \in G$ be such that $x^g = x^h$. then

$$\begin{aligned}x^g &= x^h \\x^{gh^{-1}} &= x \\gh^{-1} &\in G_x \\G_x gh^{-1} &= G_x \\G_x g &= G_x h\end{aligned}$$

Conversely, if $G_x g = G_x h$, then

$$\begin{aligned}G_x g &= G_x h \\gh^{-1} &\in G_x \\x^g &= \left(x^{(gh^{-1})}\right)^h \\&= x^h\end{aligned}$$

Thus there is a one-to-one correspondence between the points in the orbit of x and the right cosets of its stabilizer. In particular,

$$|G| = |G : G_x| |G_x| = |x^G| |G_x|.$$

2.2 Computing the Orbit of a Point

Let $\{p_i\}$ be a generating set for G . Given $x \in \{1, \dots, n\}$, we can find the orbit of x recursively:

1. Let C be an array of size n , with each element set to 0 or some other marker value initially. C will store permutations that send s to different elements of its orbit.
2. Set $C[x] = (1)$, the identity element.
3. Let Q be a queue, i.e., a list where elements are added to the end and removed from the beginning. Q will contain the points remaining to be checked.
4. Add x to Q .
5. Remove the first element from Q and call it p .
6. For each generator p_i , check to see if $C[p^{p_i}]$ has yet been defined. If not, set $C[p^{p_i}]$ to $C[p]p_i$ and add p^{p_i} to Q . Similarly, check to see if $C[p^{p_i^{-1}}]$ has yet been defined. If not, set $C[p^{p_i^{-1}}]$ to $C[p]p_i^{-1}$ and add $p^{p_i^{-1}}$ to Q .
7. Repeat Step #5 until Q is empty.

Every point stored in C is in the orbit of x by construction. The algorithm does not end, until C is closed under the action of G , so the entire orbit is stored in C . Step #5 is repeated only when new points in the orbit are found, so that the algorithm will terminate after a bounded number of sets.

C stores coset representatives for the stabilizer of x , indexed by the image of x .

2.3 Stabilizer of a Point

The stabilizer of a point $x \in \{1, \dots, n\}$ is the subgroup of G consisting of permutations that leave s fixed. Given a generating set for G , and a set C of coset representatives for G_x in G , the following theorem will help us to find a set of generators for G_x .

Theorem: (Schreier's Theorem) Let $H \subseteq G$ be a subgroup, and C be a set of coset representatives for H in G , with the identity being the coset representative for H itself. Let $\phi : G \rightarrow C$ map g to the coset representative of Hg in C . Then H is generated by the set

$$\{cg[\phi(cg)]^{-1} \mid c \in C \text{ and } g \in \{p_1, \dots, p_m\}\}.$$

Proof: Let $h = g_1g_2 \dots g_n$ be any element of H , expressed as a product of elements in G . Define $c_0 = (1)$ and $c_i = \phi(g_1 \dots g_i)$ for $i = 1, \dots, n$. Note that since $h = g_1 \dots g_n \in H$, $c_n = (1)$. Then $c_{i+1} = \phi(c_i g_{i+1})$, and we can write

$$\begin{aligned} h &= g_1 \dots g_n \\ &= c_0 g_1 \dots g_n c_n^{-1} \\ &= (c_0 g_1 c_1^{-1}) (c_1 g_2 c_2^{-1}) \dots (c_{n-1} g_n c_n^{-1}) \\ &= (c_0 g_1 [\phi(c_0 g_1)]^{-1}) (c_1 g_2 [\phi(c_1 g_2)]^{-1}) \dots (c_{n-1} g_n [\phi(c_{n-1} g_n)]^{-1}) \end{aligned}$$

and we can express any element h of H as a product of elements from $\{cg[\phi(cg)]^{-1} \mid c \in C \text{ and } g \in \{p_1, \dots, p_m\}\}$, which is therefore a generating set for H .

2.4 Method

Given a generating set of permutations, we pick a point b that is moved by at least one of them. We find the orbit of b and the corresponding coset representatives for the stabilizer G_b . We then apply Schreier's Theorem to find a set of generators for the stabilizer G_b , and we repeat the process with a new point and stabilizer, etc., until we end up with a stabilizer consisting entirely of the identity. ([1] refers to this algorithm as the Schreier-Sims method.)

Let $G = G_0 \supseteq G_1 \supseteq G_2 \supseteq \dots \supseteq G_r = \{(1)\}$ be the chain of stabilizing subgroups, with $\{b_i\}$ the set of points chosen at each stage. The collection $\{b_i\}$ is called a *base* for G : The only identity which leaves all of the b_i 's fixed is the identity. Then G_i leaves $\{b_1, \dots, b_i\}$ fixed, and is the stabilizer of b_i in G_{i-1} .

For each G_i , let $\{p_{i1}, p_{i2}, \dots\}$ be a generating set of permutations.

Let $C_i = \{c_{ij}\}$ be the set of coset representatives for G_i in G_{i-1} , where $b_i^{c_{ij}} = j$.

Suppose that $x \in G$.

1. b_1^x is in the orbit of b_1 , and there is an element $c_1 \in C_1$ such that $b_1^x = b_1^{c_1}$, i.e., such that xc_1^{-1} is in the stabilizer G_1 of b_1 .
2. $b_2^{xc_1^{-1}}$ is in the orbit of b_2 , and there is an element $c_2 \in C_2$ such that $b_2^{xc_1^{-1}} = b_2^{c_2}$, i.e., such that $xc_1^{-1}c_2^{-1}$ is in the stabilizer G_2 of b_2 .
3. Repeat, until $xc_1^{-1}c_2^{-1} \dots c_m^{-1}$ is in $G_m = \{(1)\}$. Then

$$\begin{aligned} xc_1^{-1}c_2^{-1} \dots c_m^{-1} &= (1) \\ x &= c_m c_{m-1} \dots c_2 c_1 \end{aligned}$$

Conversely, every product $c_m c_{m-1} \dots c_2 c_1$ is a product of elements of G , and thus is in G .

If $x = c_m c_{m-1} \dots c_2 c_1 = c'_m c'_{m-1} \dots c'_2 c'_1$ are two different factorizations of x , then $c_1 = c'_1$ since both are the coset representative corresponding to b_1^x . Similarly $c_2 = c'_2$ since both are the corresponding to $b_2^{xc_1^{-1}}$. By the same reasoning $c_i = c'_i$ for $i = 1, \dots, r$ and the factorizations are unique.

We now have a unique representation for every element of G that we can use to test if a given permutation $x \in G$. The size of the group is equal to the number of such representations, which in turn is the product of the sizes of C_i for $i = 1, \dots, r$, i.e., the product of the size of the orbits for each b_i .

3 Example

Let $G = G_0$ be the group generated by $p_1 = (1\ 2\ 3)(5\ 6\ 7)$ and $p_2 = (2\ 3\ 4)$.

Let $b_1 = 1$. Then we have the following orbit for 1 under p_1 and p_2 :

$$\begin{aligned} 1 &= 1^{(1)} \\ 2 &= 1^{p_1} \text{ where } p_1 = (1\ 2\ 3)(5\ 6\ 7) \\ 3 &= 1^{p_1^{-1}} \text{ where } p_1^{-1} = (1\ 3\ 2)(5\ 7\ 6) \\ 4 &= 2^{p_2^{-1}} = 1^{p_1 p_2^{-1}} \text{ where } p_1 p_2^{-1} = (1\ 4\ 3)(5\ 6\ 7) \end{aligned}$$

Let $\{c_{11}, c_{12}, c_{13}, c_{14}\}$ be the four permutations listed above. We will compute $c_{1i}p_j [\phi(c_{1i})p_j]^{-1}$ for $i = 1, 2, 3, 4$ and $j = 1, 2$. When applying ϕ , we will only need to see where $c_{1i}p_j$ maps 1. We therefore have eight generators to compute for G_1 :

1. $(1)(1\ 2\ 3)(5\ 6\ 7) [(1\ 2\ 3)(5\ 6\ 7)]^{-1} = (1)$.
2. $(1)(2\ 3\ 4) [(1)]^{-1} = (2\ 3\ 4)$.
3. $(1\ 2\ 3)(5\ 6\ 7)(1\ 2\ 3)(5\ 6\ 7) [(1\ 3\ 2)(5\ 7\ 6)]^{-1} = (1)$.
4. $(1\ 2\ 3)(5\ 6\ 7)(2\ 3\ 4) [(1\ 3\ 2)(5\ 7\ 6)]^{-1} = (2\ 4\ 3)(5\ 7\ 6)$.
5. $(1\ 3\ 2)(5\ 7\ 6)(1\ 2\ 3)(5\ 6\ 7) [(1)]^{-1} = (1)$.
6. $(1\ 3\ 2)(5\ 7\ 6)(2\ 3\ 4) [(1\ 4\ 3)(5\ 6\ 7)]^{-1} = (2\ 3\ 4)(5\ 6\ 7)$.
7. $(1\ 4\ 3)(5\ 6\ 7)(1\ 2\ 3)(5\ 6\ 7) [(1\ 4\ 3)(5\ 6\ 7)]^{-1} = (2\ 4\ 3)(5\ 6\ 7)$.
8. $(1\ 4\ 3)(5\ 6\ 7)(2\ 3\ 4) [(1\ 2\ 3)(5\ 6\ 7)]^{-1} = (1)$.

Note that most of the generators are redundant; we can omit the identity, and if we include $(2\ 3\ 4)(5\ 6\ 7)$ we do not need to include its inverse $(2\ 4\ 3)(5\ 7\ 6)$. Thus G_1 is generated by two elements: $p_2 = (2\ 3\ 4)$ and $p_3 = (2\ 3\ 4)(5\ 6\ 7)$, where we label the last permutation p_3 for convenience.

Let $b_2 = 2$. Then we have the following orbit for 2 under p_2 and p_3 :

$$\begin{aligned} 2 &= 2^{(1)} \\ 3 &= 2^{p_2} \text{ where } p_2 = (2\ 3\ 4) \\ 4 &= 2^{p_2^{-1}} \text{ where } p_2^{-1} = (2\ 4\ 3) \end{aligned}$$

Let $\{c_{22}, c_{23}, c_{24}\}$ be the three permutations listed above. We will compute $c_{2i}p_j [\phi(c_{2i})p_j]^{-1}$ for $i = 2, 3, 4$ and $j = 2, 3$. We therefore have six generators to compute for G_2 :

1. $(1)(2\ 3\ 4) [(2\ 3\ 4)]^{-1} = (1)$.
2. $(1)(2\ 3\ 4)(5\ 6\ 7) [(2\ 3\ 4)]^{-1} = (5\ 6\ 7)$.
3. $(2\ 3\ 4)(2\ 3\ 4) [(2\ 4\ 3)]^{-1} = (1)$.
4. $(2\ 3\ 4)(2\ 3\ 4)(5\ 6\ 7) [(2\ 4\ 3)]^{-1} = (5\ 6\ 7)$.
5. $(2\ 4\ 3)(2\ 3\ 4) [(1)]^{-1} = (1)$.

$$6. (2\ 4\ 3)(2\ 3\ 4)(5\ 6\ 7) [(1)]^{-1} = (5\ 6\ 7).$$

Again, most of the generators are redundant. G_3 is generated by $p_4 = (5\ 6\ 7)$.

Let $b_3 = 5$. Then we have the following orbit for 5 under p_4 :

$$\begin{aligned} 5 &= 5^{(1)} \\ 6 &= 5^{p_4} \text{ where } p_4 = (5\ 6\ 7) \\ 7 &= 5^{p_4^{-1}} \text{ where } p_4^{-1} = (5\ 7\ 6) \end{aligned}$$

Let $\{c_{55}, c_{56}, c_{57}\}$ be the three permutations listed above. We will compute $c_{5i}p_j [\phi(c_{5i}p_j)]^{-1}$ for $i = 5, 6, 7$ and $j = 4$. We therefore have three generators to compute for G_3 :

1. $(1)(5\ 6\ 7) [(5\ 6\ 7)]^{-1} = (1)$.
2. $(5\ 6\ 7)(5\ 6\ 7) [(5\ 7\ 6)]^{-1} = (1)$.
3. $(5\ 7\ 6)(5\ 6\ 7) [(1)]^{-1} = (1)$.

At this point, we see that $G_4 = \{(1)\}$ and we can stop. We know that G has size $4 \times 3 \times 3 = 36$, and we have an algorithm to see if an arbitrary permutation is contained in G .

For example, is $(1\ 2)(3\ 4) \in G$?

We start with $b_1 = 1$: $1^x = 2$, and

$$\begin{aligned} xc_{12}^{-1} &= (1\ 2)(3\ 4) [(1\ 2\ 3)(5\ 6\ 7)]^{-1} \\ &= (2\ 3\ 4)(5\ 7\ 6) \end{aligned}$$

stabilizes 1.

Next, we check $b_2 = 2$: $2^{xc_{12}^{-1}} = 3$, and

$$\begin{aligned} xc_{12}^{-1}c_{23}^{-1} &= (2\ 3\ 4)(5\ 7\ 6) [(2\ 3\ 4)]^{-1} \\ &= (5\ 7\ 6) \end{aligned}$$

stabilizes 2.

Last, we check $b_3 = 5$: $5^{xc_{12}^{-1}c_{23}^{-1}} = 7$, and

$$\begin{aligned} xc_{12}^{-1}c_{23}^{-1}c_{57}^{-1} &= (5\ 7\ 6) [(5\ 7\ 6)]^{-1} \\ &= (1) \end{aligned}$$

stabilizes 5.

Therefore $xc_{12}^{-1}c_{23}^{-1}c_{57}^{-1} = (1)$ and $x = c_{57}c_{23}c_{12}$ is in G . Note that we only needed to check three points (1, 2, and 5) in order to make that decision, as opposed to all seven points.

4 Efficiency

Note that most of the stabilizer generators that we computed were redundant, and that the number of computations can be quite large. If we were to program a computer to perform these computations without recognizing redundancy, the number of computations for each stabilizer would grow factorially. This would be bad.

A better way is to initialize all of the stabilizer subgroups to contain just the identity, and then to add G 's generators one at a time. Recalling that $G = G_0$, we would use the following procedure to add a new generator g to G_i :

1. Check to see if g is already in G_i . If so, stop.
2. Update the orbit of b_{i+1} with g added to G_i 's generators.
3. For each computation $c_i g [\phi(c_i g)]^{-1}$, add the result to G_{i+1} .
4. Apply this algorithm to each of the generators for G_{i+1} , then move on to the next generator for G_i .

5 Conclusion

By creating a chain of stabilizer subgroups, we factor a group into manageable pieces.

In the case of the group for Rubik's cube, there are 48 different squares that are moved by rotating the sides of the cubes. (6 squares in the center of each face stay fixed.) The degree is 48, and since corner faces go to corner faces and edge faces go to edge faces, each orbit has size at most 24. The orbit sizes are

$$\{24, 21, 18, 15, 12, 9, 6, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 2\}$$

for a properly chosen base, which is where the number

$$43,252,003,274,489,856,000$$

comes from.

There are many efficiency considerations that can speed up the computations, but the algorithm presented above in its plainest form permits the computation of the size (and a check for membership) of extremely large groups.

References

- [1] G. Butler, *Fundamental Algorithms for Permutation Groups*, Springer-Verlag, 1991.
- [2] John D. Dixon, Brian Mortimer, *Permutation Groups*, Springer-Verlag, 1996.
- [3] Marshall Hall, Jr., *The Theory of Groups*, Chelsea, 1976.
- [4] Nathan Jacobson, *Basic Algebra I*, W.H. Freeman, 1985.
- [5] Wilhelm Magnus, Abraham Karrass, and Donald Solitar, *Combinatorial Group Theory*, Dover, 1976.
- [6] Charles C. Sims, *Computation with Finitely Presented Groups*, Cambridge, 1994.