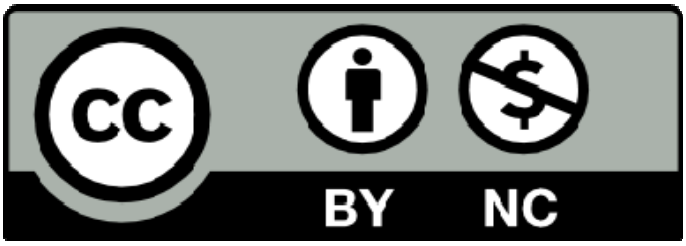
A CHILD'S GARDEN OF GROUPS

Deep Inside Rubik's Cube

(Part 8)

by

Doc Benton



Creative Commons License

You are free to:

- Share this work
- Adapt this work
- Attribute all original materials to Doc Benton

You are not free to:

- Charge money for this work; Knowledge is free!

CONTENTS (PART 8)

INTRODUCTION (PART 8)

Part 8 of this work contains an amazing amount of stuff! The focus is on *group theory* as applied to Rubik's cube, but we'll also learn in more depth about *conjugates* and *commutators* and how they apply to Rubik's cube, the *commutator* or *derived subgroup*, *centers* and *orbits*, special *subgroups* of the *Rubik's cube group*, an update on how to use GAP to study Rubik's cube, a deeper analysis of the solution to Rubik's cube, and an explanation of how to correctly count just how many permutations are possible of the *facelets* of Rubik's cube.

RUBIK'S CUBE SUBGROUPS

Some of the tools that we introduced in Part 2 and other parts of this book can now give us more information about the kinds of subgroups that exist within all the permutations that may be reached on Rubik's cube. For example, we previously stated that the total number of attainable permutations on Rubik's cube is $43,252,003,274,489,856,000$. This rather large number factors into $43,252,003,274,489,856,000 = 2^{27} \cdot 3^{14} \cdot 5^3 \cdot 7^2 \cdot 11$. Now recall that we also mentioned in Part 2 that there is an advanced theorem called the *Sylow Theorem* tells us that if a *prime number* raised to a power divides the order of a *group G*, then our *group* contains a *subgroup* composed of that many elements. In particular, there is a *subgroup* of order p^n where *p* is *prime* and p^n is the largest power of that *prime number* that divides the order of our *group G*. In this case, we call the *subgroup* of order p^n a *Sylow p-subgroup*, and *G* will also have *subgroups* of order p^m where *m* is any *nonnegative integer* less than *n*. For instance, in our *Rubik's cube group* there will be *Sylow p-subgroups* with orders of 2^{27}, 3^{14}, 5^3, 7^2, and 11. In fact, here is a generator for one of the *Sylow 11-subgroups*. Interesting, isn't it!

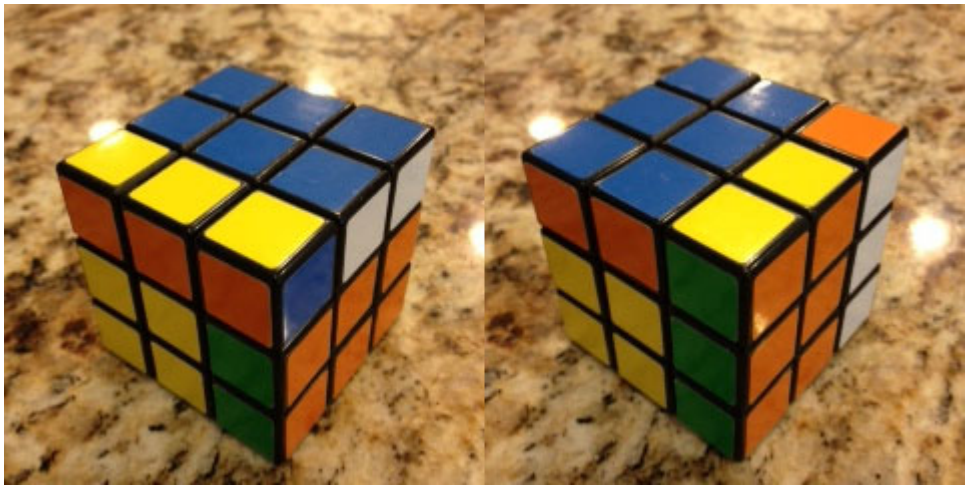
$$U^{-1}FBU^{-1}F^{-1}DBUDB^{-1}U^{-1}RRD^{-1}LLU^{-1}LLD^{-1}LLU^{-1}R$$

There will be additional *subgroups* of orders 2, 3, 5, and 7 raised to all the various powers between 1 and the power of the corresponding *Sylow p-subgroup*. And then there will undoubtedly be a whole lot of other *subgroups* whose orders are not simply a *prime* raised to a power. However, we know immediately that there is no *subgroup* of order 13. And how do we know this? Simple! It's because 13 doesn't divide the order of the *group*.

The *Rubik's cube group* itself is generated by the operations of *R*, *L*, *U*, *D*, *F*, and *B* being applied to the cube, and each individual operation generates a *cyclic group* of order 4. For example, $\langle R \rangle = \{e, R, R^2, R^3\}$ is the *subgroup* that we generate by rotating the right face of the cube a quarter turn each time, and this subgroup is isomorphic to $\mathbb{Z}_4$. Likewise, $\langle L \rangle$, $\langle U \rangle$, $\langle D \rangle$, $\langle F \rangle$, and $\langle B \rangle$ are all isomorphic to $\mathbb{Z}_4$.

Rotating the right face a quarter-turn four times

However, if we look at the *subgroup* that is generated by both *R* and *L*, i.e. by twisting the right and left faces separately, then since these operations *commute* with one another we get $\langle R, L \rangle \cong \mathbb{Z}_4 \oplus \mathbb{Z}_4$. In other words, we can think of the *group* that is generated by *R* and *L* as simply consisting of ordered pairs where the elements of *R* might occupy the first coordinate, and then the elements of *L* can occupy the second coordinate. It's a nice, uncomplicated, *abelian group*. However, if we perform the move *RU* on the cube, then we are working with cycles that overlap, and the result is far from *abelian*, or, to put it another way, $RU \neq UR$. Furthermore, if we keep repeating this move *RU*, then we eventually generate a *cyclic group* of order 105!



$RU \neq UR$

Things work out a little differently, though, if we repeatedly do the operation R^2U^2. For one thing, if we look at the cycle structure of just the permuted *cublets* and ignore any rotations or flips that might occur along the way, then we can describe the permutation created by this operation as:

$$(UB \quad UF)(BR \quad FR)(UL \quad UR \quad DR)(UBL \quad UFR \quad DBR)(ULF \quad URB \quad DRF).$$

R^2U^2

In this notation, *UB* is used to identify the *edge cublet* shared by the up face and the back face while *UFR*, for example, identifies the *corner cublet* at the up-front-right position.

Notice, though, that in the cycle structure given above that we have two cycles of length 2 and three cycles of length 3. This means that if we do this operation twice, $\left(R^2U^2\right)^2$, then we will undo the 2-*cycles* and just be left with some 3-*cycles*. In fact, the resulting permutation is

$$(DR \quad UR \quad UL)(DBR \quad UFR \quad UBL)(DRF \quad URB \quad ULF).$$

To see this algebraically, let's just raise our first permutation to the second power. If we do, then we'll get

$$\begin{aligned}
&\left[(UB \quad UF)(BR \quad FR)(UL \quad UR \quad DR)(UBL \quad UFR \quad DBR)(ULF \quad URB \quad DRF)\right]^2 \\
&= (UB \quad UF)^2 (BR \quad FR)^2 (UL \quad UR \quad DR)^2 (UBL \quad UFR \quad DBR)^2 (ULF \quad URB \quad DRF)^2 \\
&= (UL \quad UR \quad DR)^2 (UBL \quad UFR \quad DBR)^2 (ULF \quad URB \quad DRF)^2 \\
&= (DR \quad UR \quad UL)(DBR \quad UFR \quad UBL)(DRF \quad URB \quad ULF).
\end{aligned}$$

Any questions? If we look more closely at this resulting permutation, we see that it cycles three edge *cubelets* and also cycles two different sets of corner *cubelets*. In particular, the down-right, up-right, and up-left *cubelets* will cycle amongst themselves.

$(R^2U^2)^2$

If we cube R^2U^2, however, then we'll get rid of the 3-cycles and we'll be left with only a couple of 2-cycles. Algebraically, the result is

$$\begin{aligned}\left(R^2U^2\right)^3 &= \left[(UB\quad UF)(BR\quad FR)(UL\quad UR\quad DR)(UBL\quad UFR\quad DBR)(ULF\quad URB\quad DRF)\right]^3\\ &= (UB\quad UF)^3(BR\quad FR)^3(UL\quad UR\quad DR)^3(UBL\quad UFR\quad DBR)^3(ULF\quad URB\quad DRF)^3\\ &= (UB\quad UF)(BR\quad FR).\end{aligned}$$

$(R^2U^2)^3$

This final result looks particularly useful because essentially we are just swapping two back *cubelets* for two front *cublelets*, and if you try this move, then you'll you get a very nice and elegant pattern. And lastly, since R^2U^2 results in a combination of 2-*cycles* and 3-*cycles*, it follows that if we perform this operation six times, then all the *cubelets* will be restored to their original positions. When we try it, that is indeed what happens, and fortunately the orientations of the *cubelets* are also restored. Thus, the order of the *cyclic group* generated by R^2U^2 is six. In symbols, we write $\left|\left\langle R^2U^2 \right\rangle\right| = 6$. One of the very important lessons from this example, however, is that looking at the cycle structure of a permutation can help us determine not only the order of the corresponding *cyclic group*, but also what powers of this permutation might result in moving only a minimum number of *cubelets* in our Rubik's cube.

And finally, if we look at not only the *cyclic group* generated by R^2U^2, but also the *group* generated by R^2 and U^2 (denoted by $\left\langle R^2, U^2 \right\rangle$) acting either together or independently, then it turns out that this *group* has order 12 and is *isomorphic* to D_6, the symmetries of a regular hexagon. This is also an example of what on the cube we call a *two-squares group*.

Another *subgroup* of the *Rubik's cube group* that is both elegant and interesting is called the *slice group*. This *subgroup* is generated by rotating only the center slices, and as such, it will leave the corners of the cube untouched. Consequently, this group can be used to create some pretty patterns. Also, since it is not always easy to rotate a middle slice, we can accomplish the same effect by performing RL^{-1}, FB^{-1}, and UD^{-1}. Thus, the *slice group* is generated by these elements, $\left\langle RL^{-1}, FB^{-1}, UD^{-1} \right\rangle$ and $\left|\left\langle RL^{-1}, FB^{-1}, UD^{-1} \right\rangle\right| = 768$.

$$RL^{-1}, FB^{-1}, UD^{-1}$$

Also interesting and mathematically less complicated is the *slice-squared group*,

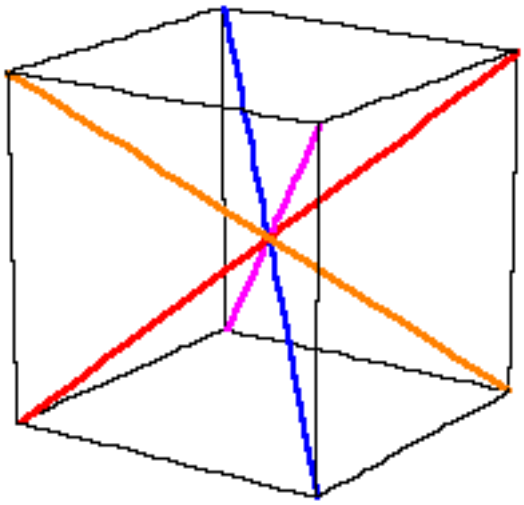
$$\left\langle (RL^{-1})^2, (FB^{-1})^2, (UD^{-1})^2 \right\rangle = \left\langle R^2L^{-2}, F^2B^{-2}, U^2D^{-2} \right\rangle = \left\langle R^2L^2, F^2B^2, U^2D^2 \right\rangle.$$

$$R^2L^2, F^2B^2, U^2D^2$$

This subgroup consists of eight elements, and it's *abelian*. And that means, by the *Fundamental Theorem of Finite Abelian Groups*, there are only three possibilities for the structure of the *slice-squared group*. It has to be *isomorphic* to either $\mathbb{Z}_8$, $\mathbb{Z}_4 \times \mathbb{Z}_2$, or $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$. See if you can figure out what the correct answer is!

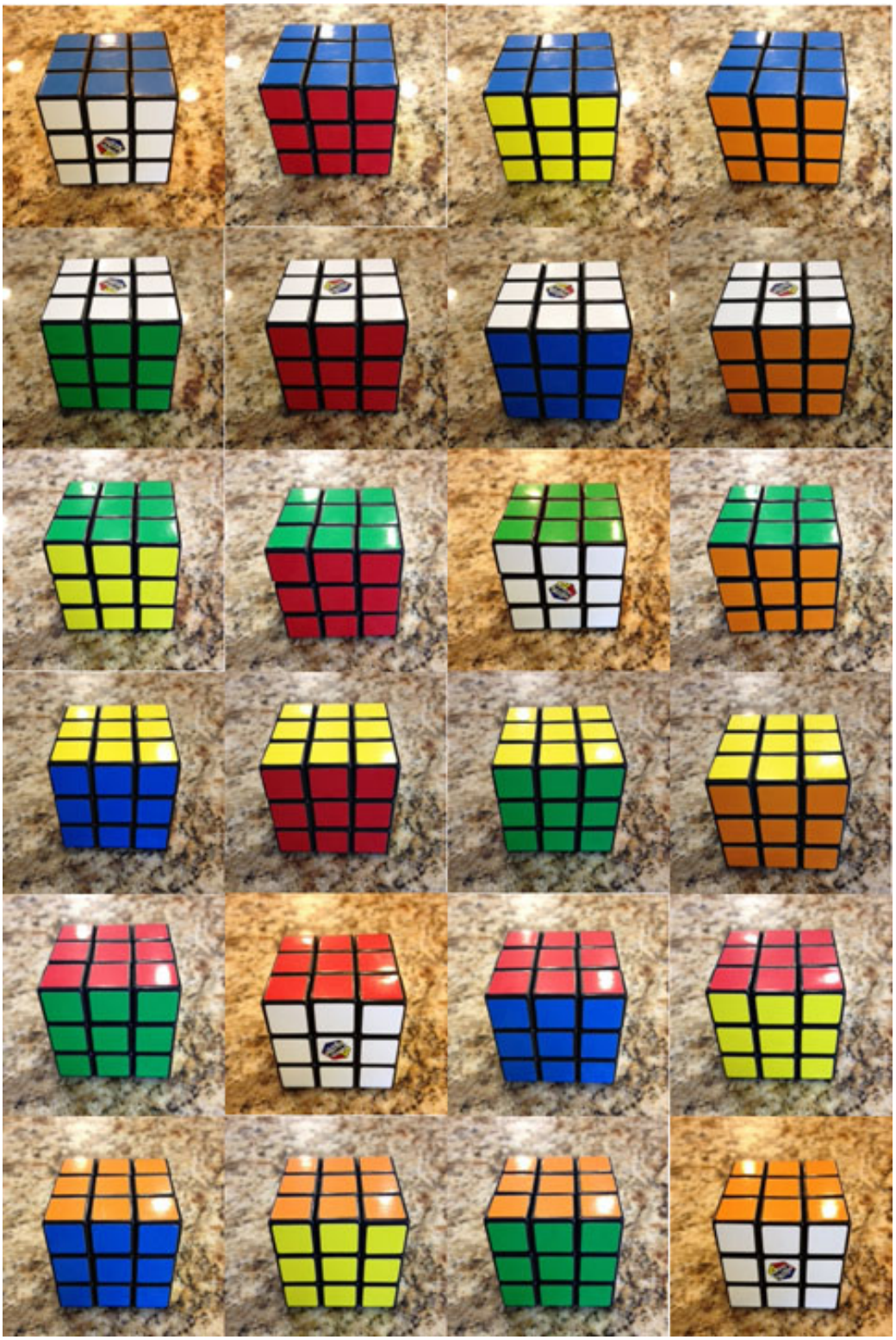
And finally, I want to talk about just one more *subgroup* that I can associate with Rubik's cube. This is going to be the *subgroup* generated by rotating the whole

cube clockwise with respect to either the up face, the right face, or the front face. I'll represent quarter turns in each of these directions by **U**, **R**, and **F**. Since these moves create a permutation of the six faces of the cube, the *group* generated has to be some *subgroup* of S_6 which has order $6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$. However, we won't get S_6 in its entirety. In fact, I claim that our *subgroup* will only have order 24. To see this, notice that we have six choices we could make regarding which colored face to have at the top of our cube. However, once we have picked a top color, then we have four choices for the front color, and once we have made these two choices, then we're done. Those two choices will establish a particular arrangement for the six faces of the cube. Thus, the total number of arrangements we can have is $6 \cdot 4 = 24$. Another way to look at this is to construct the four possible diagonals that can go from a bottom corner of the cube to a top corner of the cube, and let's suppose we give each diagonal a different color, such as red, blue, orange, or magenta.



Then every turn of the cube by **U**, **R**, or **F** will produce some permutation of these four diagonals, and the total number of permutations possible is $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$. Furthermore, notice that $\mathbf{U}^{-1}\mathbf{R}\mathbf{U}$ is equivalent to **F**. Thus, we could generate this *group* using only **U** and **R**, but it's conceptually easier to think of it as being generated by **U**, **R**, and **F**. Also, since all of the pictures below represent solved

cubes, we don't really consider this *group* to be a *subgroup* of the *Rubik's cube group*.



24 Different Orientations for the Solved Rubik's Cube

CONJUGATES, COMMUTATORS, CENTERS, AND ORBITS

Recall that if *G* is a *group*, and $x, a \in G$, then the element $a^{-1}xa$ is called a *conjugate* of *x*. Similarly, axa^{-1} is also a *conjugate* of *x* since $a^{-1} \in G$ and $(a^{-1})^{-1}xa^{-1} = axa^{-1}$. Additionally, if *H* is a *subgroup* of *G*, $H \leq G$, then we can define the *conjugate* by *a* of the whole *subgroup* as $aHa^{-1} = \{axa^{-1} \mid x \in H\}$. And now recall that if *H* is a *subgroup* of *G*, then so is aHa^{-1}. We'll prove this is the case just for finite groups since that is our primary interest. (Note that this is the first time that we are introducing you to a formal proof. Many more proofs will be done and explained in Parts 9 and 10 of this work.)

Theorem: If *G* is a finite *group*, *H* is a *subgroup* of *G*, and $a \in G$, then aHa^{-1} is also a *subgroup* of *G*.

Proof: Since *G* is a finite *group*, it suffices to show that aHa^{-1} is closed under multiplication. Thus, suppose that $b, c \in aHa^{-1}$. Then there exist *x* and *y* in *H* such that $b = axa^{-1}$ and $c = aya^{-1}$. Hence, $bc = (axa^{-1})(aya^{-1}) = a(xy)a^{-1} \in aHa^{-1}$ since $xy \in H$. Therefore, aHa^{-1} is a *subgroup* of G. □

We've mentioned previously that some *subgroups* have the special property that $aHa^{-1} = H$ for all $a \in G$, and when this happens, we say that the *subgroup* is a *normal subgroup* and write $H \lhd G$. What our theorem above shows is that even if $aHa^{-1} \neq H$, then aHa^{-1} will still be a *subgroup* of *G*. Also, remember that if the only *normal subgroups* of a *group G* are *G* and $\{e\}$, then we call *G* a *simple group*.

Recall now our earlier discussion of *Sylow p-subgroups* where our theorem said that if p^n is the highest power of a prime *p* that divides into the order of our *group G*, then *G* will have a subgroup of order p^n, a *Sylow p-subgroup*. We'll now state our second and third *Sylow Theorems*. Also, we'll defer proofs of these theorems until part 10.

The Second *Sylow Theorem*: Let *G* be a finite *group*, and let *p* be a prime that divides the order of *G*. Then all *Sylow p-subgroups* of *G* are *conjugate* to one another.

The Third Sylow Theorem: The number of *Sylow p-subgroups* of a finite *group G* is a divisor of the order of *G*. (More specifically, if $|G| = p^n \cdot m$, then the number of *Sylow p-subgroups* is a divisor of *m*.)

Hence, from this it follows that if our *Sylow p-subgroup* is not *normal*, then we can find all find all of the *Sylow p-subgroups* just by taking *conjugates* of a single *Sylow p-subgroup*. If we go back to our multiplication table for S_3, we can easily verify that all the *subgroups* of order 2 are *conjugate*.

	$(1)(2)(3)$	$(1\ \ 2)$	$(1\ \ 3)$	$(2\ \ 3)$	$(1\ \ 2\ \ 3)$	$(1\ \ 3\ \ 2)$
$(1)(2)(3)$	$(1)(2)(3)$	$(1\ \ 2)$	$(1\ \ 3)$	$(2\ \ 3)$	$(1\ \ 2\ \ 3)$	$(1\ \ 3\ \ 2)$
$(1\ \ 2)$	$(1\ \ 2)$	$(1)(2)(3)$	$(1\ \ 2\ \ 3)$	$(1\ \ 3\ \ 2)$	$(1\ \ 3)$	$(2\ \ 3)$
$(1\ \ 3)$	$(1\ \ 3)$	$(1\ \ 3\ \ 2)$	$(1)(2)(3)$	$(1\ \ 2\ \ 3)$	$(2\ \ 3)$	$(1\ \ 2)$
$(2\ \ 3)$	$(2\ \ 3)$	$(1\ \ 2\ \ 3)$	$(1\ \ 3\ \ 2)$	$(1)(2)(3)$	$(1\ \ 2)$	$(1\ \ 3)$
$(1\ \ 2\ \ 3)$	$(1\ \ 2\ \ 3)$	$(2\ \ 3)$	$(1\ \ 2)$	$(1\ \ 3)$	$(1\ \ 3\ \ 2)$	$(1)(2)(3)$
$(1\ \ 3\ \ 2)$	$(1\ \ 3\ \ 2)$	$(1\ \ 3)$	$(2\ \ 3)$	$(1\ \ 2)$	$(1)(2)(3)$	$(1\ \ 2\ \ 3)$

For example, we have three *subgroups* of order 2. Namely, $\left\{\begin{matrix}(1)(2)(3)\\(1\ \ 2)\end{matrix}\right\}$, $\left\{\begin{matrix}(1)(2)(3)\\(1\ \ 3)\end{matrix}\right\}$, and $\left\{\begin{matrix}(1)(2)(3)\\(2\ \ 3)\end{matrix}\right\}$. If we now create some *conjugates* by multiplying $\left\{\begin{matrix}(1)(2)(3)\\(1\ \ 2)\end{matrix}\right\}$ by

$(1\ \ 3)$ and $(2\ \ 3)$ [Note that each of these elements is its own inverse], then we obtain: $(1\ \ 3)\begin{Bmatrix}(1)(2)(3)\\(1\ \ 2)\end{Bmatrix}(1\ \ 3)=\begin{Bmatrix}(1)(2)(3)\\(2\ \ 3)\end{Bmatrix}$ and $(2\ \ 3)\begin{Bmatrix}(1)(2)(3)\\(1\ \ 2)\end{Bmatrix}(2\ \ 3)=\begin{Bmatrix}(1)(2)(3)\\(1\ \ 3)\end{Bmatrix}$. Thus, the other two *groups* are *conjugate* to the first, and hence, they are all *conjugate* to each other.

The second concept we want to look at again is that of a *commutator*. Recall from Part 2 that if $x, y \in G$, then the *commutator* of *x* and *y* is the product $xyx^{-1}y^{-1}$. (Additionally, $x^{-1}y^{-1}xy$, $yxy^{-1}x^{-1}$, and $y^{-1}x^{-1}yx$ are also *commutators* that can be constructed from *x* and *y*.) Notice that if *G* is an *abelian group* or if *x* and *y* commute with one another, then $xyx^{-1}y^{-1}=xx^{-1}yy^{-1}=e$, the *identity element* in *G*. On the other hand, if *x* and *y* don't *commute* with one another, but if their corresponding permutations don't have much in common, then their *commutator* probably won't result in too many changes. For example, let's suppose that $x=(1\ \ 2\ \ 3)(4\ \ 5\ \ 6)$ and $y=(6\ \ 7\ \ 8)(9\ \ 10)$. Then $x^{-1}=(6\ \ 5\ \ 4)(3\ \ 2\ \ 1)$ and $y^{-1}=(10\ \ 9)(8\ \ 7\ \ 6)$. The only item both *x* and *y* permute is 6, and their *commutator* is,

$$xyx^{-1}y^{-1}=(1\ \ 2\ \ 3)(4\ \ 5\ \ 6)(6\ \ 7\ \ 8)(9\ \ 10)(6\ \ 5\ \ 4)(3\ \ 2\ \ 1)(10\ \ 9)(8\ \ 7\ \ 6)$$
$$=(5\ \ 6\ \ 8)$$

Thus, even though our permutations don't commute, the *commutator* still undoes quite a bit of what gets moved around. Similarly, when we form a *conjugate* such as $a^{-1}xa$, there is a good chance that the last multiplication by a will undo some of the scrambling done by multiplication of a permutation *x* by a^{-1}. Consequently, as we'll see in the next chapter, both *conjugates* and *commutators* can be very helpful in developing a solution to Rubik's cube since there is a good chance that we can find some that move just a few elements of the cube while leaving the rest undisturbed.

Now let's suppose that we take all the *commutators* in our *group* and form all possible, finite products with them. This will generate a *subgroup* of *G* that we call the *commutator* or *derived subgroup*. Again, if *G* is *abelian*, then this *commutator subgroup* will simply be the identity. However, if *G* is not *abelian*, then we can think of the *commutator subgroup* as measuring how far from being *abelian* it actually is. Thus, in general, we might say that the more *abelian* the *group* is, the smaller its *commutator subgroup*, and the less *abelian* it is, the larger its *commutator subgroup*. For S_3, the *commutator subgroup* is the same as its single *Sylow 3-subgroup*, $\left\{ \begin{matrix} ( ) \\ (1 \quad 2 \quad 3) \\ (1 \quad 3 \quad 2) \end{matrix} \right\}$. Also, the *derived* or *commutator subgroup* of a *group G* is always a *normal subgroup* of *G*, and the corresponding *quotient group* is always *abelian*.

In a chapter coming up soon we will see more specifically how *conjugates* and *commutators* apply to Rubik's cube, but first we want to remind you of the definition of the *center* and also introduce the notion of the *orbit* of an element that is acted upon by a *group*. First, we'll define the *center* of a *group G* as the *subset* of all elements of *G* that commute with every other element of *G*. This *subgroup* will always be a *normal subgroup* of *G*, and for S_3, the *group* of all permutations of the set $\{1,2,3\}$, the *center* consists of just the *identity element*, $( )$. Thus, this is the only element in S_3 that *commutes* with every other element in S_3.

We'll also use S_3 to explain what we mean by the *orbit* of an element that is being acted upon by a *group*. Thus, once again, let's let our set of elements be $\{1,2,3\}$ and let's let S_3 be the *group* that is creating permutations of these elements. In this case, any element that, for example, the number 1 can be changed into is thought of as being in the same *orbit* as 1. Furthermore, since repeated application of the permutation $(1,2,3)$ can change 1 into 2 or 1 into 3, it

follows that the *orbit* of 1 under S_3 is the entire set $\{1,2,3\}$. In particular, the *group* S_3 results in only a single *orbit* for this set. In the next chapter, we'll examine both the *center* and the *orbits* of the *facelets* of Rubik's cube that are permuted by the *Rubik's cube group*, and we'll see how to compute these things and more using GAP software.

HOW TO USE GAP (PART 8)

We will begin as usual by repeating all the GAP commands with learned up to this point so that you don't have to reference earlier parts of this work, and then at the end we'll introduce in red a few GAP commands that are useful for exploring *oribits* on Rubik's cube.

1. *How can I redisplay the previous command in order to edit it?*

   Press down on the control key and then also press p. In other words, "Ctrl p".

2. *If the program gets in a loop and shows you the prompt "brk>" instead of "gap>", how can I exit the loop?*

   Press down on the control key and then also press d. In other words, "Ctrl d".

3. *How can I exit the program?*

   Either click on the "close" box for the window, or type "quit;" and press "Enter."

4. *How do I find the inverse of a permutation?*

   ```
   gap> a:=(1,2,3,4);
   (1,2,3,4)
   gap> a^-1;
   ```

(1,4,3,2)

5. *How can I multiply permutations and raise permutations to powers?*

```
gap> (1,2)*(1,2,3);
(1,3)

gap> (1,2,3)^2;
(1,3,2)

gap> (1,2,3)^-1;
(1,3,2)

gap> (1,2,3)^-2;
(1,2,3)

gap> a:=(1,2,3);
(1,2,3)

gap> b:=(1,2);
(1,2)

gap> a*b;
(2,3)

gap> a^2;
(1,3,2)

gap> a^-2;
(1,2,3)
```

```
gap> a^3;
()
gap> a^-3;
()

gap> (a*b)^2;
()

gap> (a*b)^3;
(2,3)
```

6. *How can I create a group from permutations, find the size of the group, and find the elements in the group?*

```
gap> a:=(1,2);
(1,2)

gap> b:=(1,2,3);
(1,2,3)

gap> g1:=Group(a,b);
Group([ (1,2), (1,2,3) ])

gap> Size(g1);
6

gap> Elements(g1);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]

gap> g2:=Group([(1,2),(1,2,3)]);
```

Group([(1,2), (1,2,3)])

gap> g3:=Group((1,2),(2,3,4));
Group([(1,2), (2,3,4)])

7. *How can I create a cyclic group of order 3?*

gap> a:=(1,2,3);
(1,2,3)

gap> g1:=Group(a);
Group([(1,2,3)])

gap> Size(g1);
3

gap> Elements(g1);
[(), (1,2,3), (1,3,2)]

gap> g2:=Group((1,2,3));
Group([(1,2,3)])

```
gap> g3:=CyclicGroup(IsPermGroup,3);
Group([ (1,2,3) ])
```

8. *How can I create a multiplication table for the cyclic group of order 3 that I just created?*

```
gap> ShowMultiplicationTable(g1);

*         | ()        (1,2,3)    (1,3,2)
--------+---------------------------------
()        | ()        (1,2,3)    (1,3,2)
(1,2,3)   | (1,2,3)   (1,3,2)    ()
(1,3,2)   | (1,3,2)   ()         1,2,3)
```

9. *How do I determine if a group is abelian?*

```
gap> g1:=Group((1,2,3));
Group([ (1,2,3) ])

gap> IsAbelian(g1);
true

gap> g2:=Group((1,2),(1,2,3));
Group([ (1,2), (1,2,3) ])

gap> IsAbelian(g2);
false
```

10. *What do I type in order to get help for a command like "Elements?"*

```
gap> ?Elements
```

11. *How do I find all subgroups of a group?*

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

gap> g:=Group(a,b);
Group([ (1,2,3), (2,3) ])

gap> Size(g);
6

gap> Elements(g);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]

gap> h:=AllSubgroups(g);
[ Group(()), Group([ (2,3) ]), Group([ (1,2) ]), Group([ (1,3) ]),
Group([ (1,2,3) ]), Group([ (1,2,3), (2,3) ]) ]

gap> List(h,i->Elements(i));
[ [ () ], [ (), (2,3) ], [ (), (1,2) ], [ (), (1,3) ], [ (), (1,2,3),
(1,3,2) ], [ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ] ]

gap> Elements(h[1]);
[ () ]

gap> Elements(h[2]);
[ (), (2,3) ]

gap> Elements(h[3]);
[ (), (1,2) ]

gap> Elements(h[4]);
[ (), (1,3) ]


gap> Elements(h[5]);
[ (), (1,2,3), (1,3,2) ]

gap> Elements(h[6]);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]
```

12. *How do I find the subgroup generated by particular permutations?*

```
gap> g:=Group((1,2),(1,2,3));
Group([ (1,2), (1,2,3) ])

gap> Elements(g);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]

gap> h:=Subgroup(g,[(1,2)]);
Group([ (1,2) ])

gap> Elements(h);
[ (), (1,2) ]
```

13. *How do I determine if a subgroup is normal?*

```
gap> g:=Group((1,2),(1,2,3));
Group([ (1,2), (1,2,3) ])
```

```
gap> h1:=Group((1,2));
Group([ (1,2) ])
gap> IsNormal(g,h1);

gap> h2:=Group((1,2,3));
Group([ (1,2,3) ])


gap> IsNormal(g,h2);
true
```

14. *How do I find all normal subgroups of a group?*

```
gap> g:=Group((1,2),(1,2,3));
Group([ (1,2), (1,2,3) ])

gap> Elements(g);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]

gap> n:=NormalSubgroups(g);
[ Group([ (1,2), (1,2,3) ]), Group([ (1,3,2) ]), Group(()) ]

gap> Elements(n[1]);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]

gap> Elements(n[2]);
[ (), (1,2,3), (1,3,2) ]

gap> Elements(n[3]);
[ () ]
```

15. *How do I determine if a group is simple?*

```
gap> g:=Group((1,2),(1,2,3));
Group([ (1,2), (1,2,3) ])

gap> Elements(g);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]

gap> IsSimple(g);
false


gap> h:=Group((1,2));
Group([ (1,2) ])

gap> Elements(h);
[ (), (1,2) ]

gap> IsSimple(h);
true
```

16. *How do I find the right cosets of a subset H of G?*

```
gap> g:=Group([(1,2,3),(1,2)]);
Group([ (1,2,3), (1,2) ])

gap> Elements(g);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]

gap> h:=Subgroup(g,[(1,2)]);
Group([ (1,2) ])

gap> Elements(h);
[ (), (1,2) ]

gap> c:=RightCosets(g,h);
[ RightCoset(Group( [ (1,2) ] ),()), RightCoset(Group( [ (1,2) ] ),(1,3,2)),
RightCoset(Group( [ (1,2) ] ),(1,2,3)) ]

gap> List(c,i->Elements(i));
[ [ (), (1,2) ], [ (2,3), (1,3,2) ], [ (1,2,3), (1,3) ] ]
gap> Elements(c[1]);
[ (), (1,2) ]

gap> Elements(c[2]);
[ (2,3), (1,3,2) ]


gap> Elements(c[3]);
[ (1,2,3), (1,3) ]

gap> rc:=RightCoset(h,(1,2,3));
RightCoset(Group([ (1,2) ]),(1,2,3))

gap> Elements(rc);
[ (1,2,3), (1,3) ]

gap> rc:=h*(1,2,3);
RightCoset(Group([ (1,2) ]),(1,2,3))

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

17. *How can I create a quotient (factor) group?*

```
gap> g:=Group([(1,2,3),(1,2)]);
Group([ (1,2,3), (1,2) ])

gap> Elements(g);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]

gap> n:=Group((1,2,3));
Group([ (1,2,3) ])

gap> Elements(n);
[ (), (1,2,3), (1,3,2) ]

gap> IsNormal(g,n);
true
```

```
gap> c:=RightCosets(g,n);
[ RightCoset(Group([ (1,2,3) ]),()), RightCoset(Group([ (1,2,3) ]),(2,3)) ]


gap> Elements(c[1]);
[ (), (1,2,3), (1,3,2) ]

gap> Elements(c[2]);
[ (2,3), (1,2), (1,3) ]

gap> f:=FactorGroup(g,n);
Group([ f1 ])

gap> Elements(f);
[ <identity> of ..., f1 ]

gap> ShowMultiplicationTable(f);
*                 | <identity> of ... f1
------------------+-------------------------------------
<identity> of ... | <identity> of ... f1
f1                | f1                <identity> of ...
```

18. *How do I find the center of a group?*

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

gap> b:=(2,3);
(2,3)

gap> g:=Group(a,b);
Group([ (1,2,3), (2,3) ])

gap> Center(g);
Group(())

gap> c:=Center(g);
Group(())

gap> Elements(c);
[ () ]


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

gap> b:=(1,3);
(1,3)

gap> g:=Group(a,b);
Group([ (1,2,3,4), (1,3) ])

gap> c:=Center(g);
Group([ (1,3)(2,4) ])

gap> Elements(c);
[ (), (1,3)(2,4) ]
```

19. *How do I find the commutator (derived) subgroup of a group?*

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

gap> b:=(2,3);
(2,3)

gap> g:=Group(a,b);
Group([ (1,2,3), (2,3) ])

gap> d:=DerivedSubgroup(g);
Group([ (1,3,2) ])

gap> Elements(d);
[ (), (1,2,3), (1,3,2) ]


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

gap> b:=(1,3);
(1,3)

gap> g:=Group(a,b);
Group([ (1,2,3,4), (1,3) ])

gap> d:=DerivedSubgroup(g);
Group([ (1,3)(2,4) ])

gap> Elements(d);
[ (), (1,3)(2,4) ]
```

20. *How do I find all Sylow p-subgroups for a given group?*

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

gap> b:=(2,3);
(2,3)

gap> g:=Group(a,b);
Group([ (1,2,3), (2,3) ])

gap> Size(g);
6

gap> FactorsInt(6);
[ 2, 3 ]

gap> sylow2:=SylowSubgroup(g,2);
Group([ (2,3) ])

gap> IsNormal(g,sylow2);
false

gap> c:=ConjugateSubgroups(g,sylow2);
[ Group([ (2,3) ]), Group([ (1,3) ]), Group([ (1,2) ]) ]

gap> Elements(c[1]);
[ (), (2,3) ]

gap> Elements(c[2]);
[ (), (1,3) ]

gap> Elements(c[3]);
```

```
[ (), (1,2) ]

gap> sylow3:=SylowSubgroup(g,3);
Group([ (1,2,3) ])


gap> IsNormal(g,sylow3);
true

gap> Elements(sylow3);
[ (), (1,2,3), (1,3,2) ]
```

21. *How can I create the Rubik's cube group using GAP?*

First you need to save the following permutations as a pure text file with the name rubik.txt to your C-drive before you can import it into GAP.

```
r:=(25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24);
l:=(9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35);
u:=(1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19);
d:=(41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40);
f:=(17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11);
b:=(33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27);
```

And now you can read the file into GAP and begin exploring.

```
gap> Read("C:/rubik.txt");

gap> rubik:=Group(r,l,u,d,f,b);
<permutation group with 6 generators>

gap> Size(rubik);
43252003274489856000
```

22. *How can I find the center of the Rubik's cube group?*

```
gap> c:=Center(rubik);
Group([ (2,34)(4,10)(5,26)(7,18)(12,37)(13,20)(15,44)(21,28)(23,42)(29,36)(31,4
5)(39,47) ])

gap> Size(c);
2

gap> Elements(c);
[ (), (2,34)(4,10)(5,26)(7,18)(12,37)(13,20)(15,44)(21,28)(23,42)(29,36)(31,45)
(39,47) ]
```

23. *How can I find the commutator (derived) subgroup of the Rubik's cube group?*

```
gap> d:=DerivedSubgroup(rubik);
<permutation group with 5 generators>

gap> Size(d);
21626001637244928000

gap> IsNormal(rubik,d);
true
```

24. *How can I find the quotient (factor) group of the Rubik's cube group by its commutator (derived) subgroup?*

```
gap> d:=DerivedSubgroup(rubik);
<permutation group of size 21626001637244928000 with 5 generators>

gap> f:=FactorGroup(rubik,d);
Group([ f1 ])

gap> Size(f);
2
```

25. *How can I find some Sylow p-subgroups of the Rubik's cube group?*

```
gap> Read("C:/rubik.txt");

gap> rubik:=Group(r,l,u,d,f,b);
<permutation group with 6 generators>

gap> Size(rubik);
43252003274489856000

gap> FactorsInt(43252003274489856000);
[ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 7, 7, 11 ]

gap> sylow2:=SylowSubgroup(rubik,2);
<permutation group of size 134217728 with 27 generators>

gap> sylow3:=SylowSubgroup(rubik,3);
<permutation group of size 4782969 with 14 generators>

gap> sylow5:=SylowSubgroup(rubik,5);
<permutation group of size 125 with 3 generators>

gap> sylow7:=SylowSubgroup(rubik,7);
<permutation group of size 49 with 2 generators>

gap> sylow11:=SylowSubgroup(rubik,11);
```

```
Group([ (4,36,31,39,42,12,5,21,15,13,7)(10,29,45,47,23,37,26,28,44,20,18) ])
```

```
gap> Elements(sylow11);
[ (), (4,5,36,21,31,15,39,13,42,7,12)(10,26,29,28,45,44,47,20,23,18,37),
(4,7,13,15,21,5,12,42,39,31,36)(10,18,20,44,28,26,37,23,47,45,29),
(4,12,7,42,13,39,15,31,21,36,5)(10,37,18,23,20,47,44,45,28,29,26),
(4,13,21,12,39,36,7,15,5,42,31)(10,20,28,37,47,29,18,44,26,23,45),
(4,15,12,31,7,21,42,36,13,5,39)(10,44,37,45,18,28,23,29,20,26,47),
(4,21,39,7,5,31,13,12,36,15,42)(10,28,47,18,26,45,20,37,29,44,23),
(4,31,42,5,15,7,36,39,12,21,13)(10,45,23,26,44,18,29,47,37,28,20),
(4,36,31,39,42,12,5,21,15,13,7)(10,29,45,47,23,37,26,28,44,20,18),
(4,39,5,13,36,42,21,7,31,12,15)(10,47,26,20,29,23,28,18,45,37,44),
(4,42,15,36,12,13,31,5,7,39,21)(10,23,44,29,37,20,45,26,18,47,28) ]

gap> IsNormal(rubik,sylow2);
false

gap> IsNormal(rubik,sylow3);
false


gap> IsNormal(rubik,sylow5);
false

gap> IsNormal(rubik,sylow7);
false

gap> IsNormal(rubik,sylow11);
false
```

NOTE: All of the *Sylow p-subgroups* found above have *conjugates*, but the sheer size of the *Rubik's cube group* makes it too difficult to pursue them on a typical desktop computer.

26. *How do I determine if a group is cyclic?*

```
gap> a:=(1,2,3)*(4,5,6,7);
(1,2,3)(4,5,6,7)

gap> g:=Group(a);
Group([ (1,2,3)(4,5,6,7) ])

gap> Size(g);
12

gap> IsCyclic(g);
true
```

27. *How do I create a dihedral group with 2n elements for an n-sided regular polygon?*

```
gap> d4:=DihedralGroup(IsPermGroup,8);
Group([ (1,2,3,4), (2,4) ])



gap> Elements(d4);
[ (), (2,4), (1,2)(3,4), (1,2,3,4), (1,3), (1,3)(2,4), (1,4,3,2), (1,4)(2,3) ]
```

28. *How can I express the elements of a dihedral group as rotations and flips rather than as permutations?*

```
gap> d3:=DihedralGroup(6);
<pc group of size 6 with 2 generators>

gap> Elements(d3);
[ <identity> of ..., f1, f2, f1*f2, f2^2, f1*f2^2 ]

gap> ShowMultiplicationTable(d3);
*               | <identity> of ... f1              f2              f1*f2           f2^2            f1*f2^2
----------------+-------------------------------------------------------------------------------------------\
---------
<identity> of ... | <identity> of ... f1              f2              f1*f2           f2^2            f1*f2^2
f1              | f1              <identity> of ... f1*f2           f2              f1*f2^2         f2^2
f2              | f2              f1*f2^2           f2^2            f1              <identity> of ... f1*f2
f1*f2           | f1*f2           f2^2              f1*f2^2         <identity> of ... f1              f2
f2^2            | f2^2            f1*f2             <identity> of ... f1*f2^2         f2              f1
f1*f2^2         | f1*f2^2         f2                f1              f2^2            f1*f2
<identity> of ...
```

29. *How do I create a symmetric group of degree n with n! elements?*

```
gap> s4:=SymmetricGroup(4);
Sym( [ 1 .. 4 ] )

gap> Size(s4);
24

gap> Elements(s4);
[ (), (3,4), (2,3), (2,3,4), (2,4,3), (2,4), (1,2), (1,2)(3,4), (1,2,3),
(1,2,3,4), (1,2,4,3), (1,2,4), (1,3,2),
  (1,3,4,2), (1,3), (1,3,4), (1,3)(2,4), (1,3,2,4), (1,4,3,2), (1,4,2), (1,4,3),
(1,4), (1,4,2,3), (1,4)(2,3) ]
```

30. *How do I create an alternating group of degree n with* $\frac{n!}{2}$ *elements?*

```
gap> a4:=AlternatingGroup(4);
Alt( [ 1 .. 4 ] )

gap> Size(a4);
12
gap> Elements(a4);
[ (), (2,3,4), (2,4,3), (1,2)(3,4), (1,2,3), (1,2,4), (1,3,2), (1,3,4),
(1,3)(2,4), (1,4,2), (1,4,3), (1,4)(2,3) ]
```

31. *How do I create a direct product of two or more groups?*

```
gap> g1:=Group((1,2,3));
Group([ (1,2,3) ])


gap> g2:=Group((4,5));
Group([ (4,5) ])

gap> dp:=DirectProduct(g1,g2);
Group([ (1,2,3), (4,5) ])

gap> Size(dp);
6
gap> Elements(dp);
[ (), (4,5), (1,2,3), (1,2,3)(4,5), (1,3,2), (1,3,2)(4,5) ]

gap> ShowMultiplicationTable(dp);
*           | ()          (4,5)       (1,2,3)     (1,2,3)(4,5) (1,3,2)
(1,3,2)(4,5)
------------+-------------------------------------------------------------------------
---
()          | ()          (4,5)       (1,2,3)     (1,2,3)(4,5) (1,3,2)
(1,3,2)(4,5)
(4,5)       | (4,5)       ()          (1,2,3)(4,5) (1,2,3)      (1,3,2)(4,5) (1,3,2)
(1,2,3)     | (1,2,3)     (1,2,3)(4,5) (1,3,2)      (1,3,2)(4,5) ()           (4,5)
(1,2,3)(4,5) | (1,2,3)(4,5) (1,2,3)     (1,3,2)(4,5) (1,3,2)      (4,5)        ()
(1,3,2)     | (1,3,2)     (1,3,2)(4,5) ()          (4,5)        (1,2,3)
(1,2,3)(4,5)
(1,3,2)(4,5) | (1,3,2)(4,5) (1,3,2)     (4,5)       ()          (1,2,3)(4,5) (1,2,3)
```

32. *How can I create the Quaternion group?*

```
gap> a:=(1,2,5,6)*(3,8,7,4);
(1,2,5,6)(3,8,7,4)

gap> b:=(1,4,5,8)*(2,7,6,3);
(1,4,5,8)(2,7,6,3)

gap> q:=Group(a,b);
Group([ (1,2,5,6)(3,8,7,4), (1,4,5,8)(2,7,6,3) ])

gap> Size(q);
8

gap> IsAbelian(q);
false

gap> Elements(q);
[ (), (1,2,5,6)(3,8,7,4), (1,3,5,7)(2,4,6,8), (1,4,5,8)(2,7,6,3),
(1,5)(2,6)(3,7)(4,8), (1,6,5,2)(3,4,7,8),
  (1,7,5,3)(2,8,6,4), (1,8,5,4)(2,3,6,7) ]

gap> q:=QuaternionGroup(IsPermGroup,8);
Group([ (1,5,3,7)(2,8,4,6), (1,2,3,4)(5,6,7,8) ])

gap> Size(q);
8


gap> IsAbelian(q);
false

gap> Elements(q);
[ (), (1,2,3,4)(5,6,7,8), (1,3)(2,4)(5,7)(6,8), (1,4,3,2)(5,8,7,6),
(1,5,3,7)(2,8,4,6), (1,6,3,8)(2,5,4,7),
  (1,7,3,5)(2,6,4,8), (1,8,3,6)(2,7,4,5) ]
```

33. *How can I find a set of independent generators for a group?*

```
gap> c6:=CyclicGroup(IsPermGroup,6);
Group([ (1,2,3,4,5,6) ])

gap> Size(c6);
6

gap> GeneratorsOfGroup(c6);
[ (1,2,3,4,5,6) ]


gap> d4:=DihedralGroup(IsPermGroup,8);
Group([ (1,2,3,4), (2,4) ])

gap> Size(d4);
8

gap> GeneratorsOfGroup(d4);
[ (1,2,3,4), (2,4) ]


gap> s5:=SymmetricGroup(5);
Sym( [ 1 .. 5 ] )


gap> Size(s5);
120

gap> GeneratorsOfGroup(s5);
[ (1,2,3,4,5), (1,2) ]


gap> a5:=AlternatingGroup(5);
Alt( [ 1 .. 5 ] )

gap> Size(a5);
60

gap> GeneratorsOfGroup(a5);
[ (1,2,3,4,5), (3,4,5) ]


gap> q:=QuaternionGroup(IsPermGroup,8);
Group([ (1,5,3,7)(2,8,4,6), (1,2,3,4)(5,6,7,8) ])

gap> Size(q);
8

gap> GeneratorsOfGroup(q);
[ (1,5,3,7)(2,8,4,6), (1,2,3,4)(5,6,7,8) ]
```

34. *How do I find the conjugate of a permutation in the form* $a^b = b^{-1}ab$ *?*

```
gap> a:=(1,2,3,4,5);
(1,2,3,4,5)
```

```
gap> b:=(2,4,5);
(2,4,5)


gap> a^b;
(1,4,3,5,2)

gap> b^-1*a*b;
(1,4,3,5,2)
```

35. *How do I divide up a group into classes of elements that are conjugate to one another? (Note that "conjugacy" is an equivalence relation on our group G. That means that G can be separated into nonintersecting subsets that contain only elements that are conjugate to one another.)*

```
gap> d3:=DihedralGroup(IsPermGroup,6);
Group([ (1,2,3), (2,3) ])

gap> Size(d3);
6

gap> Elements(d3);
[ (), (2,3), (1,2), (1,2,3), (1,3,2), (1,3) ]

gap> cc:=ConjugacyClasses(d3);
[ ()^G, (2,3)^G, (1,2,3)^G ]

gap> Elements(cc[1]);
[ () ]

gap> Elements(cc[2]);
[ (2,3), (1,2), (1,3) ]

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

36. *How do I input a 3x3 matrix in GAP and display in its usual rectangular format?*

```
gap> x:=[[1,2,3],[4,5,6],[7,8,9]];
[ [ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ] ]


gap> PrintArray(x);
[ [  1,  2,  3 ],
  [  4,  5,  6 ],
  [  7,  8,  9 ] ]
```

37. *How do I do arithmetic with matrices?*

```
gap> x:=[[1,2],[3,4]];
[ [ 1, 2 ], [ 3, 4 ] ]


gap> y:=[[5,6],[7,8]];
[ [ 5, 6 ], [ 7, 8 ] ]


gap> PrintArray(x+y);
[ [   6,   8 ],
  [  10,  12 ] ]


gap> PrintArray(x-y);
[ [  -4,  -4 ],
  [  -4,  -4 ] ]


gap> PrintArray(x*y);
[ [  19,  22 ],
  [  43,  50 ] ]
```

38. *How do I multiply a matrix by a number (scalar)?*

```
gap> x:=[[1,2],[3,4]];
[ [ 1, 2 ], [ 3, 4 ] ]


gap> PrintArray(x);
[ [  1,  2 ],
  [  3,  4 ] ]


gap> PrintArray(2*x);
[ [  2,  4 ],
  [  6,  8 ] ]


gap> PrintArray(x/2);
[ [  1/2,    1 ],
  [  3/2,    2 ] ]
```

39. *How do I find the inverse of a matrix?*

```
gap> x:=[[1,2],[3,4]];
[ [ 1, 2 ], [ 3, 4 ] ]


gap> PrintArray(x);
[ [  1,  2 ],
  [  3,  4 ] ]


gap> xinverse:=x^-1;
[ [ -2, 1 ], [ 3/2, -1/2 ] ]
```

```
gap> PrintArray(xinverse);
[ [    -2,     1 ],
  [   3/2,  -1/2 ] ]


gap> xinverse:=1/x;
[ [ -2, 1 ], [ 3/2, -1/2 ] ]


gap> PrintArray(xinverse);
[ [    -2,     1 ],
  [   3/2,  -1/2 ] ]


gap> PrintArray(x*xinverse);
[ [  1,  0 ],
  [  0,  1 ] ]
```

40. *How do I find the transpose of a matrix?*

```
gap> x:=[[1,2],[3,4]];
[ [ 1, 2 ], [ 3, 4 ] ]


gap> PrintArray(x);
[ [  1,  2 ],
  [  3,  4 ] ]


gap> xtranspose:=TransposedMat(x);
[ [ 1, 3 ], [ 2, 4 ] ]


gap> PrintArray(xtranspose);
[ [  1,  3 ],
  [  2,  4 ] ]
```

41. *How do I find the determinant of a matrix?*

```
gap> x:=[[1,2],[3,4]];
[ [ 1, 2 ], [ 3, 4 ] ]


gap> PrintArray(x);
[ [  1,  2 ],
  [  3,  4 ] ]


gap> DeterminantMat(x);
-2
```

42. *How do I find the orbits that the Rubik's cube group creates on the set* $\{1,2,3,\ldots,48\}$ *?*

In *Windows*, use *Notepad* to type the following file, and save it to your *C*-drive.

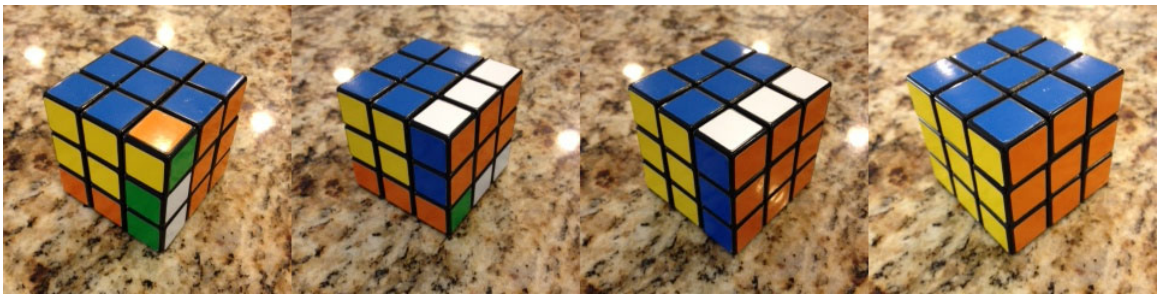
```
r:=(25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24);
l:=(9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35);
u:=(1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19);
d:=(41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40);
f:=(17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11);
b:=(33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27);
```

Now enter the following commands.

```
gap> Read("C:/rubik.txt");
gap>

gap> rubik:=Group(r,l,u,d,f,b);
<permutation group with 6 generators>


gap> Orbit(rubik,1);
[ 1, 17, 3, 14, 41, 9, 19, 38, 8, 22, 48, 40, 43, 11, 33, 46, 24, 6, 30, 27, 16,
35, 25, 32 ]


gap> Orbit(rubik,2);
[ 2, 5, 13, 18, 36, 37, 42, 39, 34, 12, 10, 31, 15, 7, 4, 26, 20, 45, 21, 44,
47, 28, 29, 23 ]


gap> o:=Orbits(rubik);
[ [ 1, 17, 3, 14, 41, 9, 19, 38, 8, 22, 48, 40, 43, 11, 33, 46, 24, 6, 30, 27,
16, 35, 25, 32 ],
[ 2, 5, 12, 36, 7, 10, 47, 45, 34, 4, 28, 13, 44, 29, 21, 26, 37, 20, 42, 15,
31, 23, 18, 39 ] ]


gap> Size(o);
2


gap> Elements(o);
[ [ 1, 17, 3, 14, 41, 9, 19, 38, 8, 22, 48, 40, 43, 11, 33, 46, 24, 6, 30, 27,
16, 35, 25, 32 ],
[ 2, 5, 12, 36, 7, 10, 47, 45, 34, 4, 28, 13, 44, 29, 21, 26, 37, 20, 42, 15,
31, 23, 18, 39 ] ]
```

CONJUGATES, COMMUTATORS, CENTERS, AND ORBITS IN RUBIK'S CUBE

Many people have either written or talked about how the cube can mostly be solved using either *conjugates* or *commutators*, and so let's look at a few ways in which we are already using them. First, at the very beginning when I am trying to correctly place the corners on the top face of the cube, I often do the move $R^{-1}DR$ which is a *conjugate*. Thus, let's think about what this move does for us. My goal with this move is to place something in the up-front-right (UFR) corner. To do this, I position the *cubelet* that I want to move there in the down-front-left (DFL) corner. Then I do R^{-1}. That moves the up-front-right (UFR) corner *cubelet* to the down-front-right (DFR) position. Next, I do D, and this moves my *cubelet* from DFL to DFR. And finally, I do R, and that rotates my *cublet* from the down-front-right (DFR) position back into the up-front-right (UFR) corner that was my goal. In a nutshell, you can say that we shifted things from the top face to a workspace down below, moved something into the workspace, and then moved it back to the top row.



$R^{-1}DR$

When we do a *commutator* on Rubik's cube, the idea is that we are, most of the time, partially undoing what we have previously done. In particular, what happens when we do a *commutator* like $R^{-1}D^{-1}RD$ on the cube is that some of

the *cubelets* get moved around, but others stay right where they are, and anytime we move just a few *cubelets*, that gives us a tool we can use for easily solving the cube.

Now let's examine some more of the algorithms we have used for solving Rubik's cube. When solving the middle layer of the cube, we used $URU^{-1}R^{-1}U^{-1}F^{-1}UF$ and $U^{-1}F^{-1}UFURU^{-1}R^{-1}$. Notice that both of these are products of *commutators*. The first part of $URU^{-1}R^{-1}U^{-1}F^{-1}UF$ is the *commutator* $URU^{-1}R^{-1}$, and the second part is the *commutator* $U^{-1}F^{-1}UF$. Likewise with $U^{-1}F^{-1}UFURU^{-1}R^{-1}$ the first part is the *commutator* $U^{-1}F^{-1}UF$ and the second part is the *commutator* $URU^{-1}R^{-1}$.

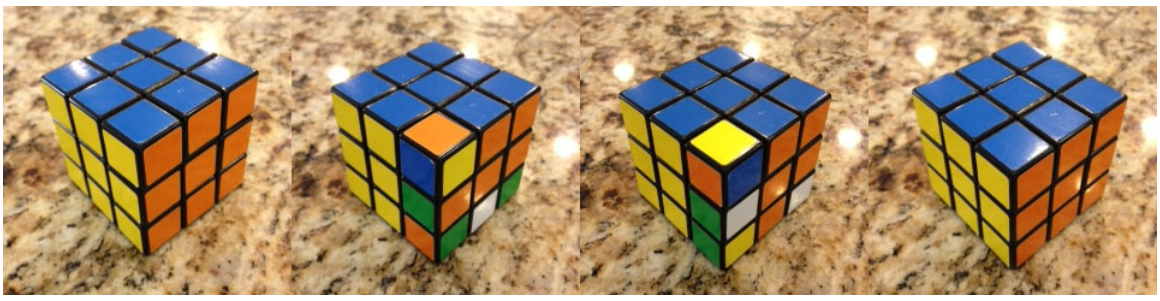
When we move on to the top layer of the cube, we apply the algorithm $FRUR^{-1}U^{-1}F^{-1}$. And now if we look at this algorithm more closely, we can see that we are really just taking the *conjugate* of a *commutator*. In other words, the inner part of this algorithm is the *commutator* $RUR^{-1}U^{-1}$, and then we *conjugate* this by *F* to get $F\left(RUR^{-1}U^{-1}\right)F^{-1}$.

The next algorithm we apply to the top layer is $URU^{-1}L^{-1}UR^{-1}U^{-1}L$. If we split this up into URU^{-1} and $L^{-1}UR^{-1}U^{-1}L = (L^{-1}U)R^{-1}(U^{-1}L) = (U^{-1}L)^{-1}R^{-1}(U^{-1}L)$, then we can easily see that both of these movements are *conjugates*. Also, if we were to remove the rotations of the top face from $URU^{-1}L^{-1}UR^{-1}U^{-1}L$, then we would be left with the *commutator* $RL^{-1}R^{-1}L$.

And now, let's examine our final algorithm $\left(R^{-1}D^{-1}RD\right)^2$ in greater detail. At the core of this particular algorithm is the *commutator* $R^{-1}D^{-1}RD$, and $\left(R^{-1}D^{-1}RD\right)^2$ is the move that we usually use at the end to get our final corner *cubelets* turned correctly. If we do this move just once, then we'll transpose two sets of *corner cubelets*, and we'll cycle three *edge cubelets*,

$$(DB \quad DR \quad FR)(DRF \quad UFR)(DBR \quad DLB).$$

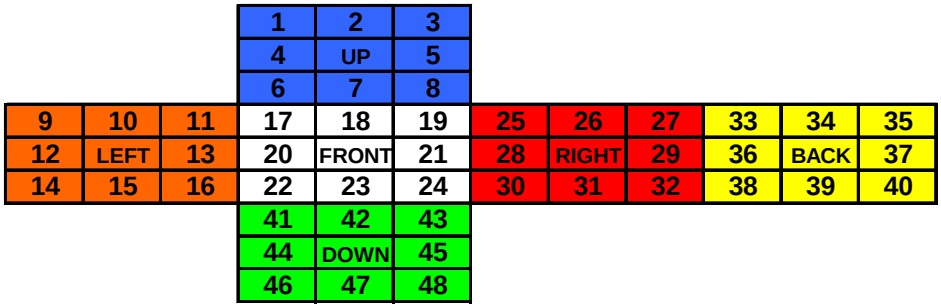
Thus, if we repeat this operation a second time, then we'll restore the *corner cubelets* and just cycle the *edge cubelets*. However, when I do this, my *corner cubelets*, while being in the right corners, have also been twisted clockwise through an angle of $120°$. And as you might suspect from the presence of the 3 & 2-cycles that we have above for $R^{-1}D^{-1}RD$, repeating this move 6 times, the least common multiple of 2 and 3, will finally restore everything back to its starting point. And now you can see why this is our finishing move for the cube. At the end, we have all the *cubelets* in their correct positions, but some of the *corner cubelets* on top are usually twisted. To untwist them, we apply the algorithm $(R^{-1}D^{-1}RD)^2$ until we get one corner untwisted. Then we rotate the top to move another twisted corner into position, and we repeat with $(R^{-1}D^{-1}RD)^2$ until that one is untwisted. However, take my word for now that the one thing that we are mathematically guaranteed is that the number of times we have to do $(R^{-1}D^{-1}RD)^2$ is always going to be some multiple of 3. Thus, suppose we have to do $(R^{-1}D^{-1}RD)^2$ just three times. Then $[(R^{-1}D^{-1}RD)^2]^3 = (R^{-1}D^{-1}RD)^6 = e$. In other words, since our algorithm has order 6, by the time we are done untwisting the *cubelets*, everything has been returned to its proper position and orientation.



$(R^{-1}D^{-1}RD)^2, (R^{-1}D^{-1}RD)^4, (R^{-1}D^{-1}RD)^6$

Using GAP software, we can easily explore facets of Rubik's cube that would be hard to examine just by hand. For example, we can easily find information about

what we call the *center* of the *Rubik's cube group* as well as the *orbits* of the *facelets* on the cube that it permutes. First, though, let's recall how to initially set up GAP for exploring Rubik's cube. The initial step is to give each *facelet* of the cube a number so that we can express moves in terms of permutations of those numbers. Back in Part 2 of this work we did it as follows, and from that we were able to define the moves for right, left, up, down, front, and back as permutations.




```
r:=(25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24);
l:=(9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35);
u:=(1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19);
d:=(41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40);
f:=(17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11);
b:=(33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27);
```

The next step is to copy the permutations above and save them as a text file, not to your *documents* folder, but to your *C*-drive as a file named *rubik.txt*. Also, in my old version of *Windows*, for this to work correctly I have to create the file using *Notepad* instead of *Wordpad*. It appears that if you try to save this as a text file using *Wordpad*, then some extra structure is also saved that interferes with reading the file into GAP. However, once you have correctly saved your file, then you can open it in GAP by typing the following command.

```
gap> Read("C:/rubik.txt");
gap>
```

And now you can create the group *rubik* by typing in the following.

```
gap> rubik:=Group(r,l,u,d,f,b);
<permutation group with 6 generators>
```

```
gap> Size(rubik);
43252003274489856000
```

And now, to find the *center* of the *Rubik's cube group*, just type in the following.

```
gap> c:=Center(rubik);
Group([ (2,34)(4,10)(5,26)(7,18)(12,37)(13,20)(15,44)(21,28)(23,42)(29,36)(31,4
5)(39,47) ])

gap> Size(c);
2

gap> Elements(c);
[ (),
(2,34)(4,10)(5,26)(7,18)(12,37)(13,20)(15,44)(21,28)(23,42)(29,36)(31,45)(39,47
) ]
```

From this we can see that there are just two elements in the *Rubik's cube group* that commute with every other element in the *group*. One of these elements is the *identity*, and the other is a long chain of *transpositions*,

```
(2,34)(4,10)(5,26)(7,18)(12,37)(13,20)(15,44)(21,28)(23,42)(29,36)(31,45)(39,47)
```

And by the way, the permutation above is called the *superflip*. Its effect is that it flips the colors of each *edge cublet* on the cube. Thus, if an *edge cublet* is colored green-white in the cube's solved state, then those two colors are flipped so that green becomes white and white becomes green. Below is a picture of the cube illustrating the *superflip*, and this is followed by two different algorithms for creating the *superflip*.

$$UR^2FBRB^2RU^2LB^2RU^{-1}D^{-1}R^2FR^{-1}LB^2U^2F^2$$

or

$$FLULB^{-1}U^{-1}D^{-1}LF^{-1}U^{-1}B^{-1}RL^{-1}BF^2U^{-1}D^{-1}F^2B^2R^2U^{-1}D^{-1}$$

The *center* is a *normal subgroup* of a *group*, and if we look at the corresponding *quotient group*, then the *center* of the *quotient* will consist of only the *identity element* of that *group.* Also, since the size of the *Rubik's cube group* is

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

and since the size of the *center* of the *Rubik's cube group* is 2, it follows that the size of resulting *quotient group* is exactly half the size of the complete *Rubik's cube group*.

$$43,252,003,274,489,856,000/2 = 21,626,001,637,244,928,000$$

Now let's examine the *orbits* associated with the *Rubik's cube group*. First, notice from our diagram above where we assigned a number to each *facelet* that the *Rubik's cube group* creates permutations of the *facelets* that we have numbered 1 through 48. In other words, the *Rubik's cube group* acts upon the set $\{1,2,3,\ldots,48\}$. Also, notice that half of the numbers in this set correspond to *facelets* on *corner cublets*, and the other half correspond to *facelts* on *edge cubelets*. To find the *orbit* of any of these numbers, just enter the following into GAP.

```
gap> Orbit(rubik,1);
[ 1, 17, 3, 14, 41, 9, 19, 38, 8, 22, 48, 40, 43, 11, 33, 46, 24, 6, 30, 27, 16,
35, 25, 32 ]
```

```
gap> Orbit(rubik,2);
[ 2, 5, 13, 18, 36, 37, 42, 39, 34, 12, 10, 31, 15, 7, 4, 26, 20, 45, 21, 44,
47, 28, 29, 23 ]
```

And to find all the *orbits* that the *Rubik's cube group* creates in our set $\{1,2,3,\ldots,48\}$, type in the following.

```
gap> o:=Orbits(rubik);
[ [ 1, 17, 3, 14, 41, 9, 19, 38, 8, 22, 48, 40, 43, 11, 33, 46, 24, 6, 30, 27,
16, 35, 25, 32 ], [ 2, 5, 12, 36, 7, 10, 47, 45, 34, 4, 28, 13, 44, 29, 21, 26,
37, 20, 42, 15, 31, 23, 18, 39 ] ]

gap> Size(o);
2

gap> Elements(o);
[ [ 1, 17, 3, 14, 41, 9, 19, 38, 8, 22, 48, 40, 43, 11, 33, 46, 24, 6, 30, 27,
16, 35, 25, 32 ], [ 2, 5, 12, 36, 7, 10, 47, 45, 34, 4, 28, 13, 44, 29, 21, 26,
37, 20, 42, 15, 31, 23, 18, 39 ] ]
```

From the above we see that our set has just two *orbits*, and if we examine them more closely, then we see that the numbers in the first *orbit* correspond to *facelets* on *corner cublets* while the numbers in the second *orbit* correspond to *facelets* on *edge cublets*.

And now since we have talked above about *commutators* with regard to the solution for Rubik's cube, let's use GAP to find the *commutator* or *derived subgroup* and also the corresponding *quotient group*. To find the *derived subgroup*, type in the following.

```
gap> d:=DerivedSubgroup(rubik);
<permutation group with 5 generators>

gap> Size(d);
21626001637244928000
```

Since the size of the *derived* or *commutator subgroup* is half the size of the *Rubik's cube group*, it follows that the corresponding *quotient group* (also known as a *factor group*) has size 2, and hence, the *quotient group* is *isomorphic* to C_2, the *cyclic group* of order 2. Also, GAP has a single command for finding this *factor group*.

```
gap> q:=CommutatorFactorGroup(rubik);
Group([ f1 ])


gap> Size(q);
2


gap> Elements(q);
[ <identity> of ..., f1 ]


gap> ShowMultiplicationTable(q);
*                 | <identity> of ... f1
------------------+-----------------------------------
<identity> of ... | <identity> of ... f1
f1                | f1                <identity> of ...
```

PATTERNS ON RUBIK'S CUBE

We now just want to examine a few interesting patterns you can create on the surface of Rubik's cube. However, our interest goes beyond just art. There are certainly many places where one can go and instantly find algorithms for all sorts of patterns for the cube, but we want to do more than that. As usual, we want to explain some of the math that comes with these patterns.

1. This first pattern is one of my favorites. In this case, the algorithm simply switches two front edge *cubelets* with the corresponding back *edge cubelets*. It's simple, but elegant. Also, this algorithm generates a *cyclic group* of order 2, C_2, and that means that the algorithm is its own inverse.

$\left(R^2U^2\right)^3$

2. This next pattern is created using elements of the *slice group*. The *slice group* can be thought of as generated by moving only the center slices of the cube, and thus, the *corner cubelets* stay fixed. Consequently, a lot of nice patterns can be created using only slices. Also, even though we conceptualize this group in terms of moving center slices, moves such as UD^{-1}, RL^{-1}, and FB^{-1} accomplish the same thing. In the pattern below we

have 6 dots centered on backgrounds of different colors. Notice that this algorithm has order 3 and so the *cyclic group* generated is isomorphic to C_3.

$UD^{-1}RL^{-1}FB^{-1}UD^{-1}$

3. This next pattern of six checkerboards comes from the *slice squared group* that is generated by U^2D^2, R^2L^2, and F^2B^2. Also, recall that this *group* is *abelian*. Consequently, the above three moves may be done in any order. Furthermore, the *slice squared group* has order 8 and is *isomorphic* to $C_2 \times C_2 \times C_2$, while the *cyclic group* generated by $U^2D^2R^2L^2F^2B^2$ is a *subgroup* of the *slice squared group* that has order 2 and is, consequently, *isomorphic* to C_2.

$U^2D^2R^2L^2F^2B^2$

4. In this pattern, we've taken the previous pattern and created a *conjugate* of the form xyx^{-1}, and with luck, this will also transform one interesting pattern into another. In this case, we see our previous pattern of six checkerboards transformed into one of four checkerboards. Furthermore, this algorithm generates a *cyclic group* of order 2 which is *isomorphic* to C_2.

$$\left(R^2U^2\right)^3U^2D^2R^2L^2F^2B^2\left(R^2U^2\right)^3$$

5. The next four pictures are going to be based on patterns from the *slice squared group*. We'll first look at a pattern from this group, and then we'll form a conjugate of that pattern using $\left(R^2U^2\right)^3$. Also, both algorithms below generated *cyclic groups* of order 2, C_2. (Recall that $\left(R^2U^2\right)^3 = \left(R^2U^2\right)^{-3}$.)

F^2B^2

$\left(R^2U^2\right)^3 F^2B^2 \left(R^2U^2\right)^3$

6. This time we'll start with $R^2L^2F^2B^2$ and then, again, form a conjugate with $\left(R^2U^2\right)^3$, in other words, $\left(R^2U^2\right)^3 R^2L^2F^2B^2\left(R^2U^2\right)^3$. Also, once again both of these algorithms will generate *cyclic groups* that are *isomorphic* to C_2.

$R^2L^2F^2B^2$

$\left(R^2U^2\right)^3 R^2L^2F^2B^2\left(R^2U^2\right)^3$

7. This one is one of my favorites. It creates a center dot on two of the faces, a checkerboard on two, and stripes on the remaining two. And once again the *cyclic groups* generated are *isomorphic* to C_2.

$$\left(R^2U^2\right)^3 R^2L^2F^2B^2\left(L^2U^2\right)^3$$

8. If you start with the red face in front and the white face to the right, then this will create 4 crosses with one of them a red Templar cross on a white background. I like this one because a few of my ancestors on my dad's side were Knights Templar. Also, the algorithm below generates a *cyclic group* of order 4, C_4, and both the square and the cube of the algorithm generate similar cross patterns.

$$LUFLULDLDU^{-1}F^{-1}U^{-1}F^{-1}D^{-1}F^{-1}L^{-1}D^{-1}F^{-1}$$

9. Again start with the red face in front and the white fact to the right. This algorithm will produce 6 crosses with one of them a red Templar cross on a white background. This algorithm has order 3, C_3, and the square of the algorithm also results in 6 crosses.

$L^2R^{-1}FD^2L^{-1}F^{-1}RL^{-1}FB^{-1}LFU^2L^{-1}F^2B$

10. One thing I like to do is to see what new patterns I can create by combining $UD^{-1}RL^{-1}FB^{-1}UD^{-1}$ with $\left(R^2U^2\right)^3$. See what interesting things you can come up with!

Start with $UD^{-1}RL^{-1}FB^{-1}UD^{-1}$, rotate the whole cube, and do $UD^{-1}RL^{-1}FB^{-1}UD^{-1}$ again to get a pattern of four dots.

Take the 4-dot pattern above, rotate the cube so the white face is front with red on top, and add $\left(R^2U^2\right)^3$. Rotate the whole cube again, and then do $\left(R^2U^2\right)^3$ again. You should now have a 4-dot pattern combined with a checkerboard pattern!

11. A favorite pattern of many cube enthusiasts is the cube within a cube. The lengthy algorithm below generates a *cyclic group* of order 3, C_3. Additionally, if you do the algorithm a second time, then you get another cube within a cube, and if you do it three times, then the cube is restored to the solved state.

$$FLFU^{-1}RUF^{2}L^{2}U^{-1}L^{-1}BD^{-1}B^{-1}L^{2}U$$

12. And finally, this algorithm creates an incredible pattern called the *superflip*. Basically, every *cubelet* is in its home position, but every single *edge cubelet* has been flipped. It's pretty easy to see that the *group* generated by this move has order 2, C_2, but what is not so obvious is that this element of the *Rubik's cube group* commutes with every other element of that *group*. In fact, the only other element in the *group* that does that is the *identity*. In *group theory*, the set of all elements of a *group* that commute with every other element is called the *center* of the *group*, and the *center* of the *Rubik's cube group* consists of only the *identity* and the *superflip*.

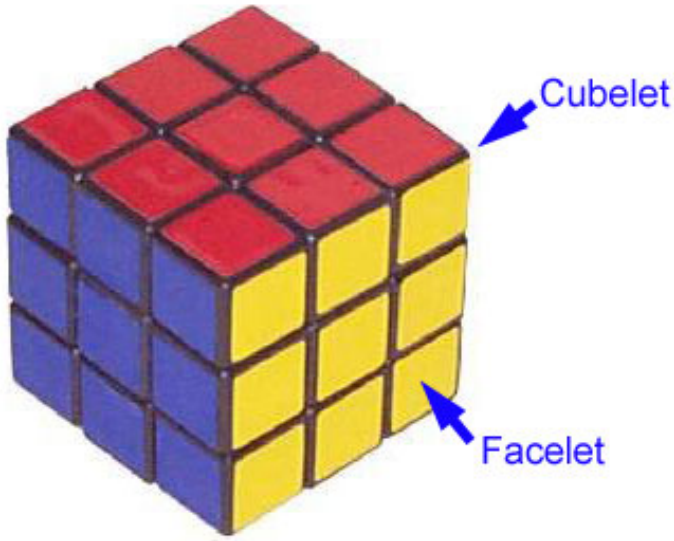
$$UR^2FBRB^2RU^2LB^2RU^{-1}D^{-1}R^2FR^{-1}LB^2U^2F^2$$

or

$$FLULB^{-1}U^{-1}D^{-1}LF^{-1}U^{-1}B^{-1}RL^{-1}BF^2U^{-1}D^{-1}F^2B^2R^2U^{-1}D^{-1}$$

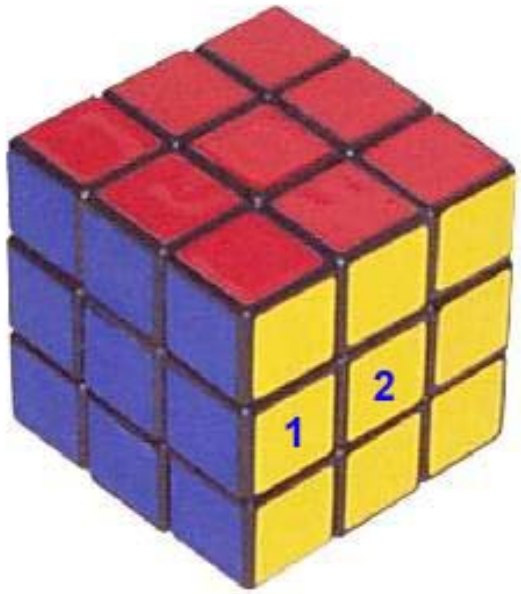
Counting the Number of Permutations in Rubik's Cube

Below is a picture of Rubik's cube. The surface reveals 26 smaller cubes that we'll call "*cubelets*[1]" and 54 smaller faces that we'll call "*facelets*."

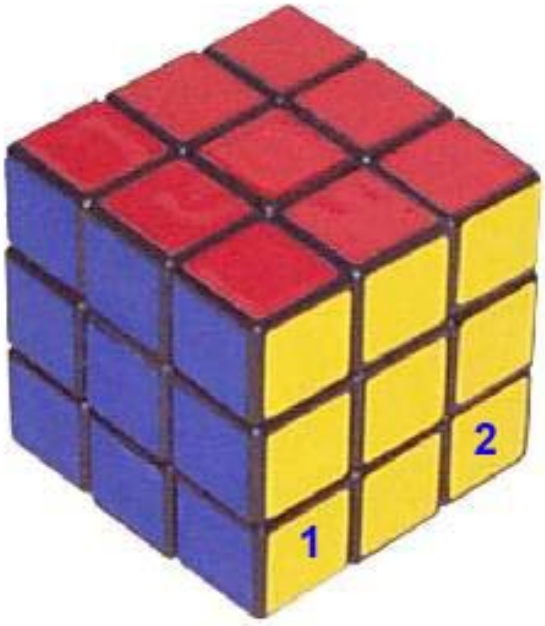


At first glance, you might think that the total number of permutations we can make of the *facelets* on Rubik's cube is $54! \approx 2.3 \times 10^{71}$, the number of permutations we can make of 54 things, but this is going to give us a number that is way too large. It's too large because we can't take a single *facelet* and just move it anywhere. There are going to be some restrictions on where *facelets* can wind up. For example, suppose we number a couple of the *facelets* as below.

[1] Many people also refer to "*cubelets*" as "*cubies*."



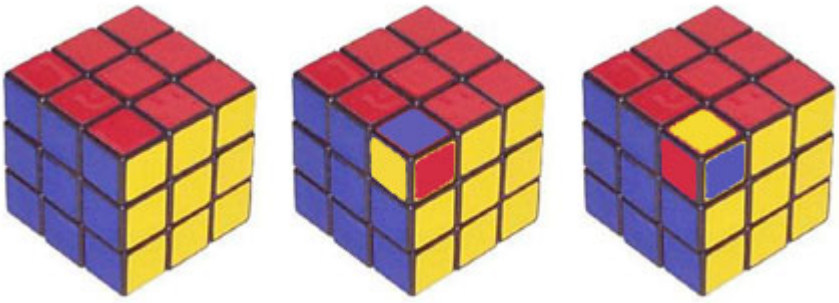
Then there is no way that we can rotate the sides of the cube to make these numbers wind up in the following positions.



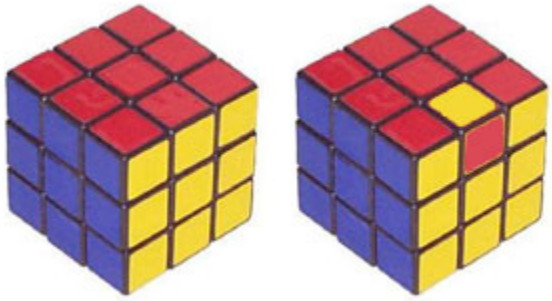
And why can't we do this? It's because we have three types of *cubelets* – *center cubelets*, *edge cubelets*, and *corner cubelets*. Furthermore, every time we rotate

a face of the cube, the *center cubelet* stays where it is, an *edge cubelet* just gets moved to the position of another *edge cubelet*, and a *corner cubelet* gets moved to another *corner*. Thus, since our original numbers 1 & 2 begin on an *edge* and a *center cubelet*, respectively, they can never wind up on *corner cubelets*. Additionally, we'll sometimes us notations like UF and UFR to refer, respectively, to the *edge cubelet* in the up-front position and the *corner cubelet* in the up-front-right position.

At this point, you might notice that the *facelets* of a single *cubelet* always have to stay together, and thus, maybe the total number of possible permutations of the *facelets* of Rubik's cube will just be equal to the number of permutations of the 26 *cubelets* or $26! \approx 4.03 \times 10^{26}$. Well, this is still going to be too large a number because, again, there are restrictions on where you can move *center*, *edge*, and *corner cubelets*. As we just mentioned, every time we rotate a face, the *center cubelet* stays where it is, a *corner cubelet* replaces another *corner cubelet* and an *edge cubelet* replaces an *edge cubelet*. Thus, to count the actual number of possible permutations, perhaps we need to begin by multiplying the number of permutations you can make from the 8 *corner cubelets* times the number of permutations you can make from the 12 *edge cubelets*. This gives us $(8!)(12!) \approx 1.9 \times 10^{13}$. However, there are a couple of things we haven't taken into consideration yet. One is that each *corner cubelet* can be rotated among three different positions, and the other is each *edge cubelet* can be flipped back and forth from one position to another. These rotations and flips are illustrated by the pictures below.



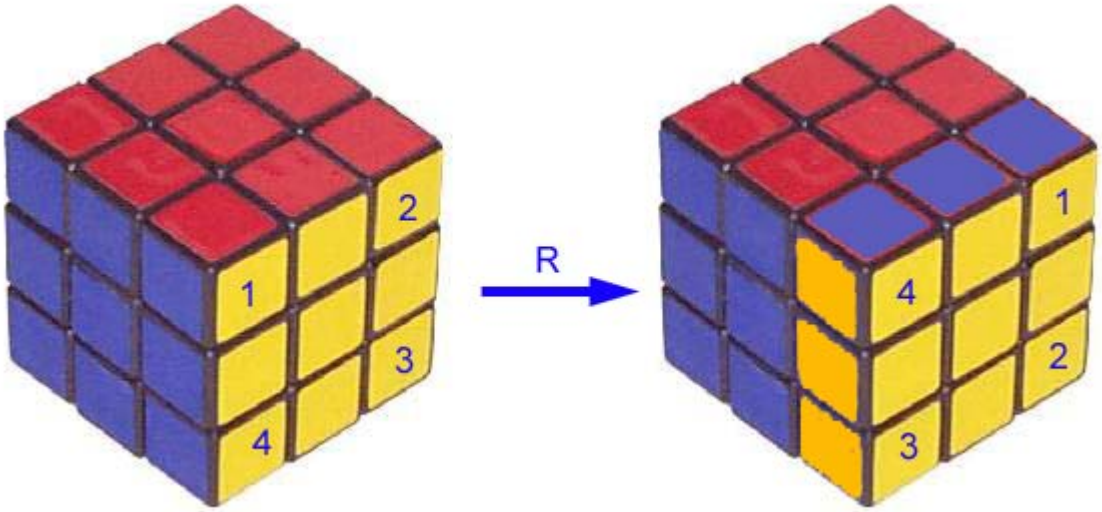
Rotations of a corner cubelet



Flipping an edge cubelet

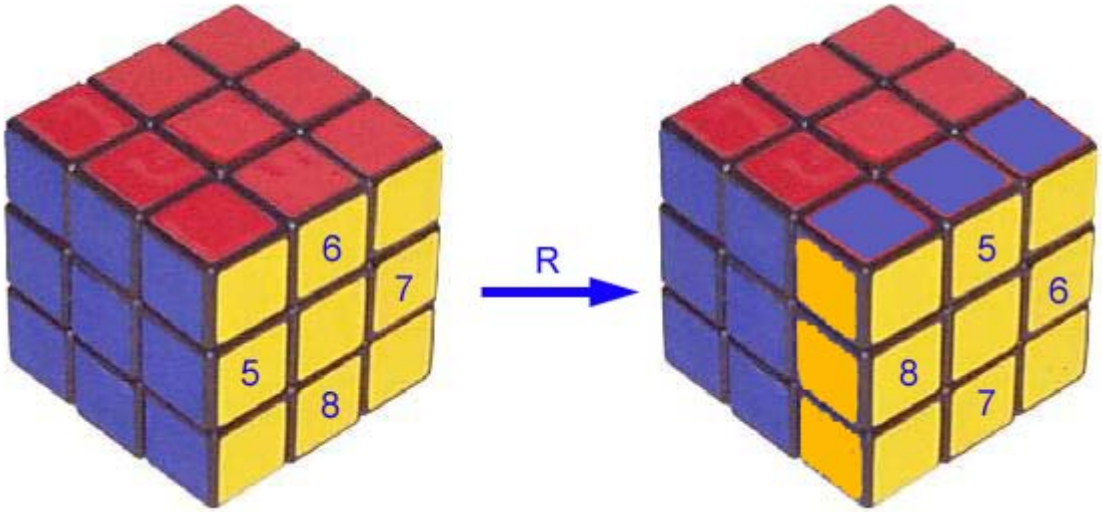
Thus, each of the eight *corner cubelets* could be in any of three rotational states, and so we should multiply our previous number by 3^8. Similarly, since each of the twelve *edge cubelets* could be in either of two states, flipped or not flipped, we should also multiply our previous estimate by 2^{12}. This will give us $(8!)(12!)(3^8)(2^{12}) \approx 5.2 \times 10^{20}$. This is smaller than our previous estimate of $26! \approx 4.03 \times 10^{26}$, but still too large, and so let's see what we can do to reduce it.

First off, let's number the *corner cubelets* 1 through 4 on the right face of the cube, and then let's see what kind of permutation results when we rotate the right face a quarter-turn clockwise.

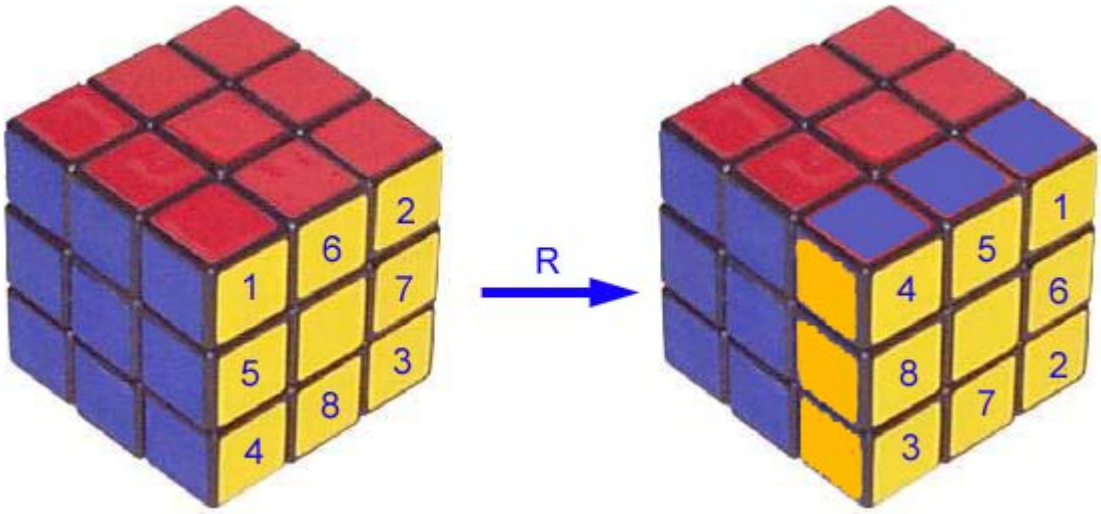


We can express this permutation as $(1\ \ 2\ \ 3\ \ 4) = (1\ \ 2)(1\ \ 3)(1\ \ 4)$, and thus, we see that it is an odd permutation since it can written as a product of three transpositions.

Now let's number the *edge cubelets* 5 through 8 and do the same clockwise rotation of the right face.



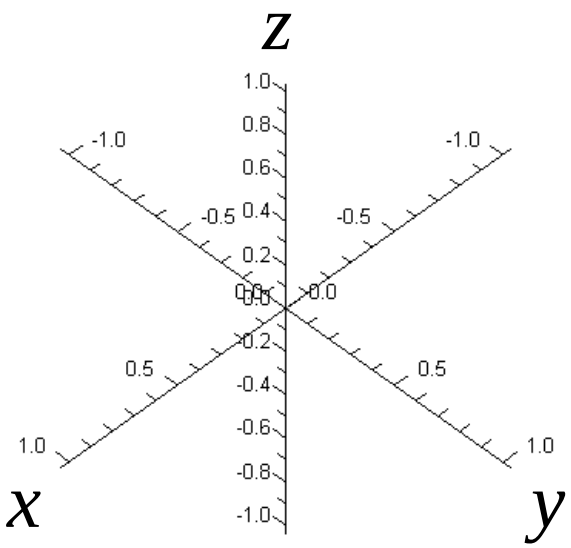
We can express this result as $(5\ \ 6\ \ 7\ \ 8)=(5\ \ 6)(5\ \ 7)(5\ \ 8)$, and once again we get an odd permutation. However, if we now consider the permutations of the *corner* and *edge cubelets* together, then the final result of our clockwise quarter-turn is an even permutation consisting of six transpositions.

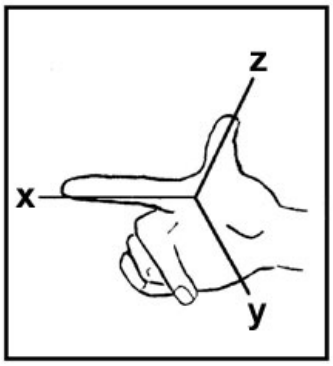


$$(1\ \ 2\ \ 3\ \ 4)(5\ \ 6\ \ 7\ \ 8)=(1\ \ 2)(1\ \ 3)(1\ \ 4)(5\ \ 6)(5\ \ 7)(5\ \ 8)$$

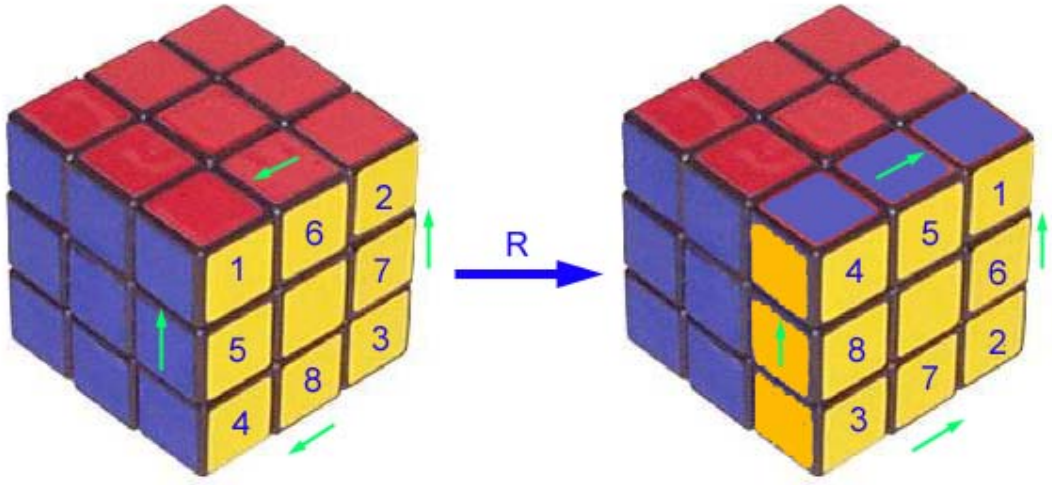
At this point, what this means is that every turn of a face of a cube results in an even permutation, and, hence, any combination of turns will also result in an even permutation. Thus, the number of possible permutations of the *cubelets* in Rubik's cube is not $(8!)(12!)(3^8)(2^{12})$. Instead, it is no more than half of this, $\frac{(8!)(12!)(3^8)(2^{12})}{2}$, since only half of the permutations represented by the number $(8!)(12!)(3^8)(2^{12})$ are even. However, this is still not our final answer. There are more things to consider!

To see what else we need to take into account, let's begin with a typical representation of the coordinate axes in three dimensional space using what is known as a right-handed coordinate system.



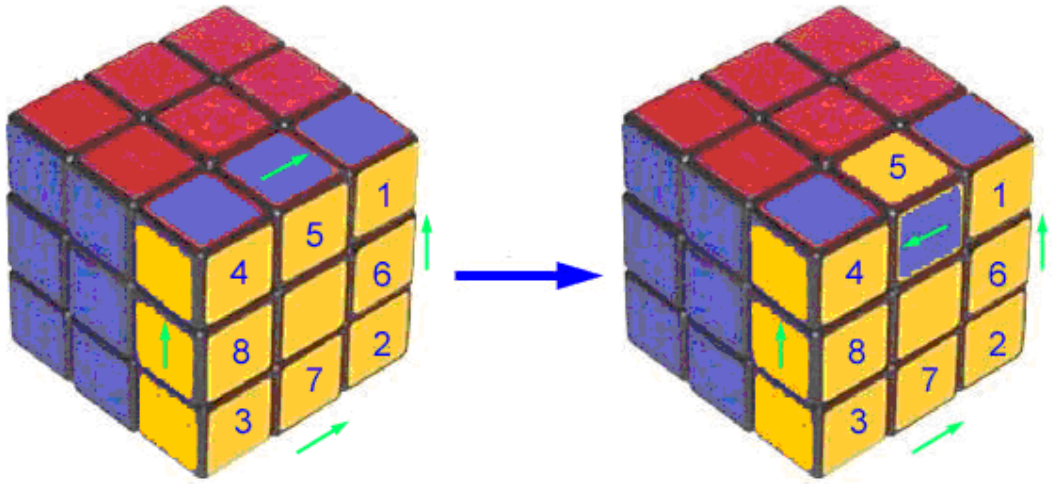


In the diagram above, the axes are labeled on the positive side. Now let's suppose that we attach arrows to the *edge cubelets* on a side of Rubik's cube such that the arrows are pointing either in the direction of positive *x* or positive *z*. And finally, let's once again rotate the right face of our cube a quarter-turn in the clockwise direction, and let's see what happens to our arrows.



The end result is that two of the arrows are now pointing in the direction of negative *x*. However, we could also say that the overall orientation is still positive

since the product (positive)(negative)(positive)(negative) = positive. In particular, we can never wind up, after turning the face of a cube a quarter-turn, with an orientation such as (positive)(positive)(positive)(negative) = negative. Notice that this orientation would also correspond to a single *edge cubelet* being flipped.

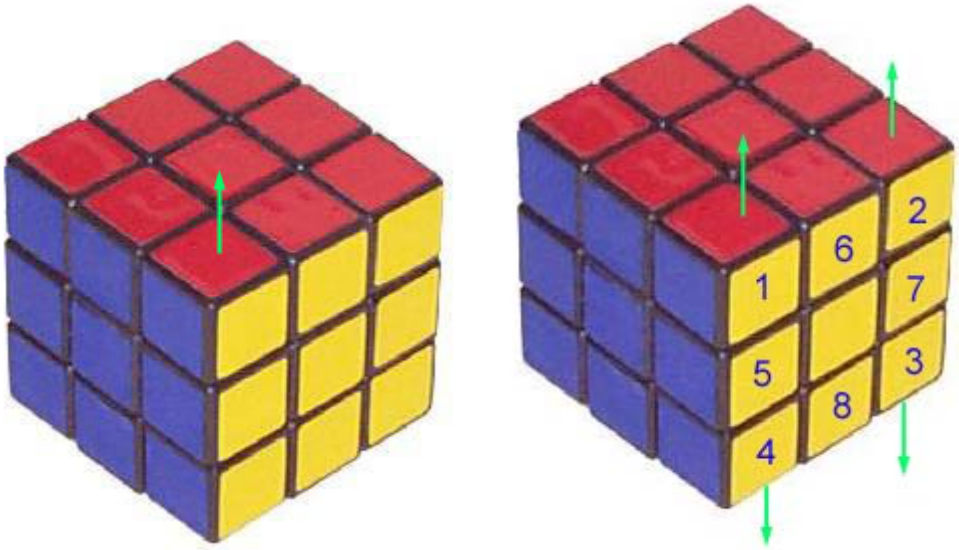


Flipping an *edge cubelet* changes its orientation

Thus, since every quarter-turn of a face leaves us with a positive orientation, so will any combination of turns of the faces of Rubik's cube. In particular, the number of "flipped" *edge cubelets* always has to be even. And as far as our problem of counting the number of permutations of Rubik's cube goes, this means that we have to divide our last number by 2 again since only half of that number will correspond to the positive orientations of *edge cubelets* that we have just defined. Thus, the number of permutations that we can achieve is now no more than $\frac{(8!)(12!)(3^8)(2^{12})}{2 \cdot 2} \approx 1.30 \times 10^{20}$.

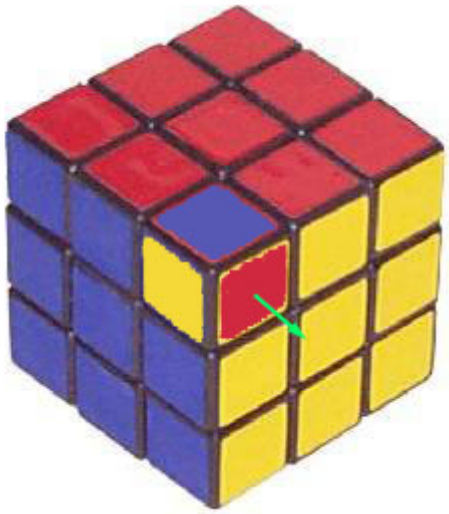
There's just one more thing we have to consider, and then we'll be done. In particular, we need to consider how rotating a face of the cube might twist or rotate a *corner cubelet*. For example, below I've attached an arrow to the top *facelet* of the red-yellow-blue *corner cublet*. If I now do a sequence of rotations

of the faces of the cube such that when I'm done the *cubelet* is either on the top face with arrow is pointing up or on the bottom face with the arrow pointing down, then I'll consider the *cubelet* to have not been rotated.

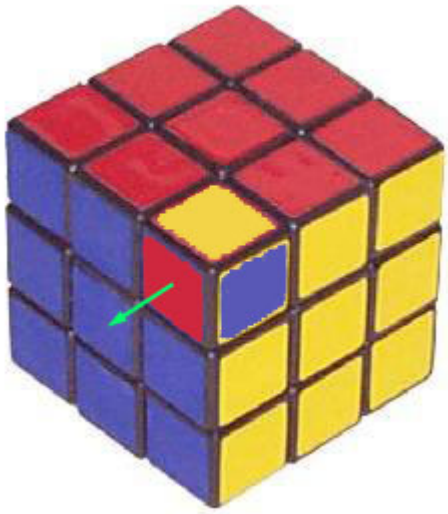


.

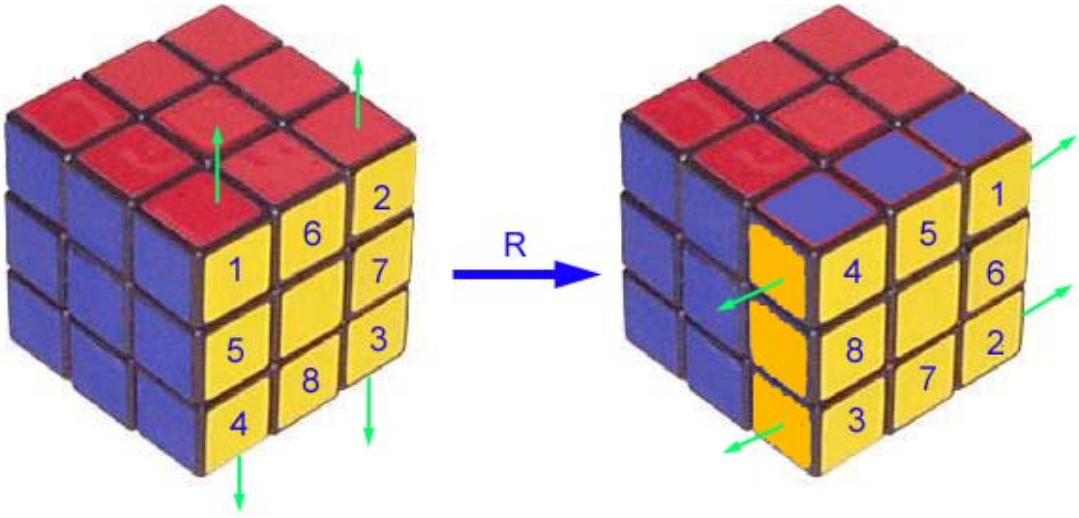
On the other hand, if I wind up with something like the image below, then I'll say that the *cubelet* has been rotated clockwise through an angle of $120°$.



And finally, if I wind up with the following image, then I'll say that my red-yellow-blue *cubelet* has been rotated clockwise through an angle of $240°$



And now we're good to go! First, it should be evident that if all I do is rotate the top face or the bottom face of the cube, then none of the *corner cubelets* will undergo any rotation whatsoever. However, if we rotate any of the side faces (right, left, front, or back), then it's a different story. Below I've placed some arrows on the c*orner cubelets* of the right face and then rotated the right face a quarter-turn clockwise.



If we look at the *corner cubelet* that I've labeled 1, then it has not only been moved to a new position, it has also been rotated through an angle of $120°$. In particular, the blue *facelet* is now on top instead of the red. Likewise, the *corner cubelet* labeled 2 has been moved from the top face to the bottom face, but instead of having the red *facelet* on the bottom, the *cubelet* appears to have been rotated clockwise through an angle of $240°$. And similarly, we could say that the *cubelet* labeled 3 has been rotated clockwise through an angle of $120°$, and the *cubelet* labeled 4 has been rotated clockwise through an angle of $240°$[2]. If we now add up total number of degrees of rotation for each of the *corner cublets*, it's clear that the sum has to be either a whole number multiple of $360°$ or a multiple of $360°$ plus an additional $120°$ or a multiple of $360°$ plus an additional $240°$. In the first instance, we'll say that the cube has orientation 1, in the second case that it has orientation 2, and in the third case that it has orientation 3.

Well, when we rotate the right face a quarter-turn clockwise as we did above, the sum of the angles of rotation for the *corner cubelets* is $120° + 240° + 120° + 240° = 720° = 2 \cdot 360°$. Thus, the cube is left in orientation 1. Furthermore, the sum of the sum of the angles of rotation along each side is $360°$. And now, a moment's reflection or experimentation should convince you that if you rotate any other face of the cube or any combination of faces of the cube, then the final orientation is still going to be 1. However, since there are three conceivable orientations that the cube could be left in, orientation 1 represents only a third of them, and that means that only one-third of the *corner cubelet* configurations that I had previously counted are actually attainable. Thus, if we divide our previous calculation by 3, then we will obtain the true number of permutations that can be made of the *facelets* on Rubik's cube. The result is slightly more than forty-three quintillion.

[2] Since *cubelet* 4 is now on top, a rotation of $0°$ would correspond to the arrow pointing up, but instead, it's pointing in the direction corresponding to a $240°$ rotation.

$$\frac{(8!)(12!)(3^8)(2^{12})}{2\cdot 2\cdot 3} = 43,252,003,274,489,856,000 = 2^{27}3^{14}5^3 7^2 11$$

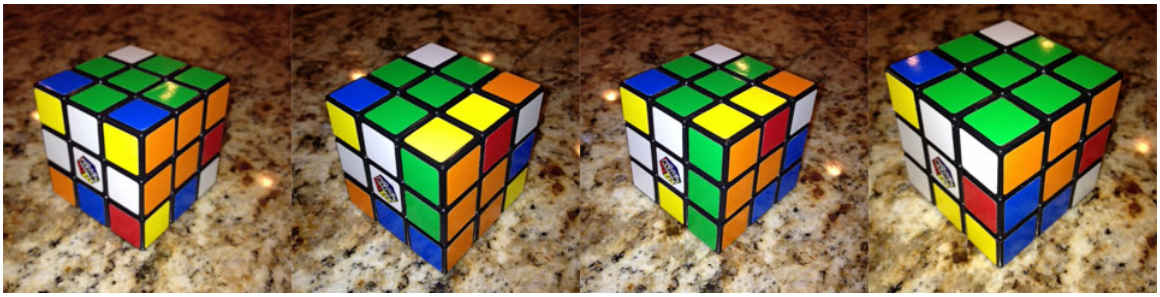
Notice that if we could move any *corner cubelet* to any *corner* position and any *edge cubelet* to any *edge* position, then the correct number of possible permutations would be $(8!)(12!)(3^8)(2^{12})$. However, what we have just shown is that only a twelfth of these permutations are actually attainable. Thus, if you take your cube apart and start randomly reassembling it, then you have only a 1 in 12 chance of creating a cube that can be restored to its original configuration. And finally, how do we know that we still haven't overcounted the number of permutations? Simple. Because Chuck Norris has actually done all $43,252,003,274,489,856,000$ permutations![3]

[3] Chuck Norris has also counted to infinity twice, and Chuck Norris CAN divide by zero. I, on the other hand, have only counted to infinity once, but I did start at infinity and count down.

REVISITING THE SOLUTION TO RUBIK'S CUBE

Now that we know a lot about the mathematics behind Rubik's cube, it's time to take a closer at the solution we use. The first part, of course, is pretty easy. When I'm trying to solve Rubik's cube, I always begin with the green center *cubelet* on top so that I can finish the green face first. I begin with the goal of initially completing the green cross on top, and that is really very easy. I simply rotate faces until I get the green *facelet* of an edge cube positioned on the down face of the cube. I then rotate the down face until the other color on the edge cube matches the center cube. And finally, I rotate the appropriate face $180°$ to bring the green *facelet* to the up face. I then repeat until I've finished my green cross, and I don't really have to formalize the procedure too much since I'm not that worried, yet, about what's going on with the rest of the cube.

Once I've completed the green cross on the up face, I still improvise quite a bit to get the corners positioned. However, I do usually make use of the maneuvers $R^{-1}DR$ and $FD^{-1}F^{-1}$ in order to get my *corner cubelet* placed with the right orientation. Sometimes, though, the green *facelet* of a *corner cubelet* is on the down face of the cube, and when this happens I may do something like $DFD^{-1}F^{-1}R^{-1}D^{-1}R$ to rotate it in the bottom layer.



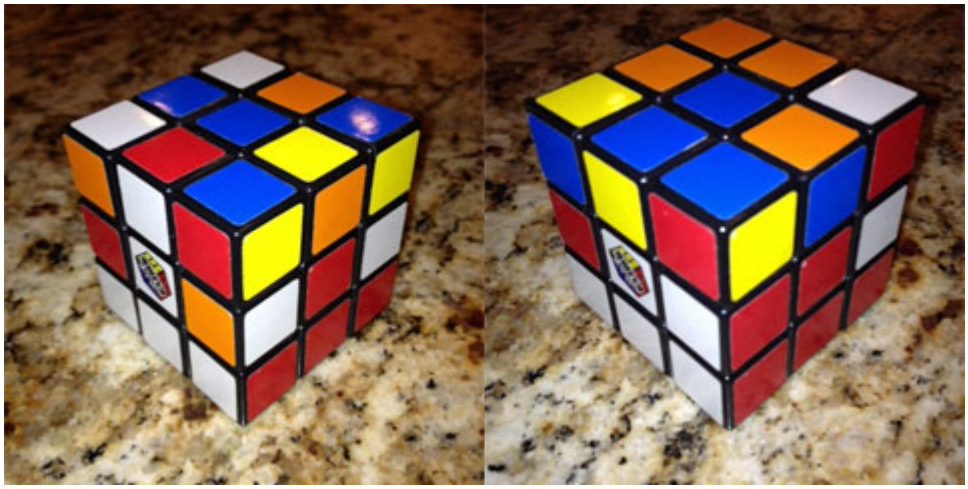
$R^{-1}DR$

Once the green face is completed, I turn the cube over so that green is on the down face, and then I proceed in a systematic way to place the *edge cubelets* in the middle layer. The two algorithms that are used in our solution are $URU^{-1}R^{-1}U^{-1}F^{-1}UF$ and $U^{-1}F^{-1}UFURU^{-1}R^{-1}$. What should be clear at this point is that both algorithms are products of *commutators* which means that they belong to the *commutator subgroup* of the *Rubik's cube group*. Also, as we have seen previously, *commutators* have a tendency to move only a few elements. Given that, let's look at the first algorithm in a bit more detail. If we perform only the first part, $URU^{-1}R^{-1}$, then the resulting permutation given as a product of cycles can be written as $(UB\ \ UR\ \ FR)(DRF\ \ UFR)(UBL\ \ URB)$ where a notation like *UB* refers to the up-back *edge cubelet* and notation like *UBL* refers to the up-back-left *corner cubelet*. Also, we can now see the promise of this permutation. It contains a 3-*cycle* that involves two *edge cubelets* on the up face and the *edge cubelet* in the front-right position. Just what we want! It also involves a couple of 2-*cycles* that move *corner cubelet*s, and one of them only switches *corner cubelets* on the up face. Unfortunately, the other one switches the up-front-right *corner cubelet* with the down-right-front *corner cublet*, and that will mess up the green face that we just completed. However, if we perform our algorithm twice, $\left(URU^{-1}R^{-1}\right)^2$, then the result is just the 3-*cycle* $(FR\ \ UR\ \ UB)$, and that looks promising except for the fact that this also twists the down-right-front *corner cublet* into a different orientation. Thus, let's see how the permutation $U^{-1}F^{-1}UF$ might fix things for us. If we do this algorithm, then the resulting cycle structure is

$(UL \quad UF \quad FR)(DRF \quad UFR)(ULF \quad UBL)$. What we immediately see is that our algorithm will once again permute two *edge cubelets* in the up face with the front-right *edge cubelet* and it will also switch the down-right-front *corner cubelet* with the up-front-right *corner cubelet*. Exactly what we need! Furthermore, when we multiply the two permutations together we get:

$(UB \quad UR \quad FR)(DRF \quad UFR)(UBL \quad URB)(UL \quad UF \quad FR)(DRF \quad UFR)(ULF \quad UBL)$

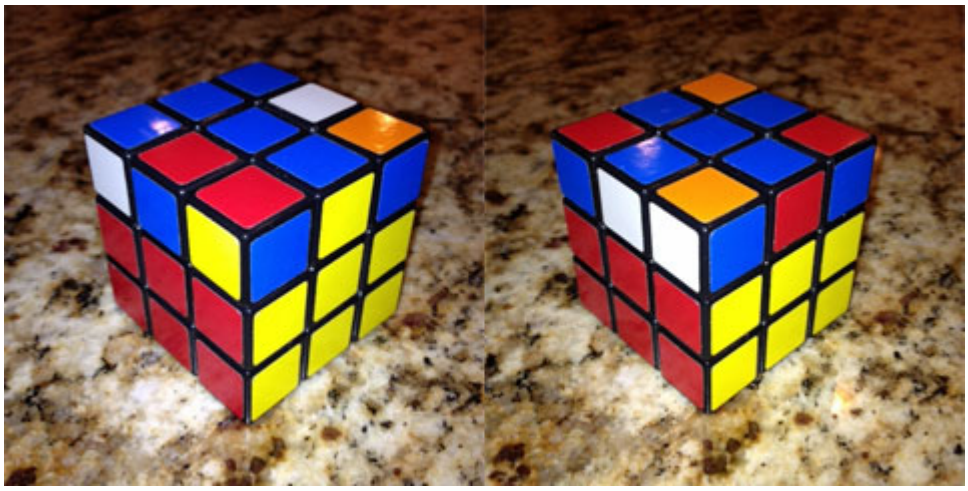
$= (UB \quad UR \quad UL \quad UF \quad FR)(UBL \quad URB \quad ULF)$. From this result we can see that the up-front *edge cubelet* moves into the front-right position, and everything else that happens is basically a permutation of *cubelets* on the up face. Absolutely perfect! Furthermore, when we try it out, we see that the *facelet* on the front of the up-front *edge cubelet* remains on the front face when it is moved to the front-right position. This means that our algorithm will work fine just so long as the *cubelet* that we want to move to the middle layer has the right *facelet* on the front face, but if its orientation is flipped, then that's why we need our second algorithm, $U^{-1}F^{-1}UFURU^{-1}R^{-1}$. If we look at the cycle structure for this one, then we get $(UL \quad UF \quad UB \quad UR \quad FR)(ULF \quad URB \quad UBL)$. For this one, we need to first get the *cubelet* we want to place moved into the up-right position, and then our algorithm will move it to the front-right position with the proper orientation. Well, this argument may be a little hard to follow, and consequently, I recommend going through these algorithms yourself using either a Rubik's cube or free Rubik's cube software that can generally be found online.



$URU^{-1}R^{-1}U^{-1}F^{-1}UF$

The next step in our solution to Rubik's cube is to get the blue *facelets* on the up face for all of the *edge cublets* on our top layer, and we can achieve this with the help of the *commutator* $RUR^{-1}U^{-1}$. However, when we look at the corresponding cycle structure, we see that this *commutator* moves an *edge cubelet* on the up face to the front-right position, $(FR \quad UR \quad UB)(DRF \quad UFR)(UBL \quad URB)$. Fortunately, there's an easy fix for this. Simply begin by turning the front face clockwise (*F*) before doing your *commutator*, and then turn it back again counterclockwise (F^{-1}) when you are done. In other words, do $FRUR^{-1}U^{-1}F^{-1}$. This maneuver will constrain all the movement to the top face, $(UB \quad UF \quad UR)(ULF \quad UFR)(UBL \quad URB)$. When we perform this algorithm, we'll also see that the up-left *cubelet* never moves, and that two of our *edge cubelets* get flipped as we go from up-back to up-front and up-front to up-right. Consequently, often all we have to do is to simply repeat this algorithm until all the *edge cubelets* have the proper *facelet* on the up face, and if that doesn't work, then you may have to throw in a rotation of the up face in between applications of the algorithm. Something else to notice is that if we perform this algorithm three times, then the resulting cycle structure is going to be $(ULF \quad UFR)(UBL \quad URB)$. In other words, we switch the two front *corner cubelets* on top with each other and we also switch the two back *corner cubelets*. Additionally, when we actually perform this maneuver, we see that the

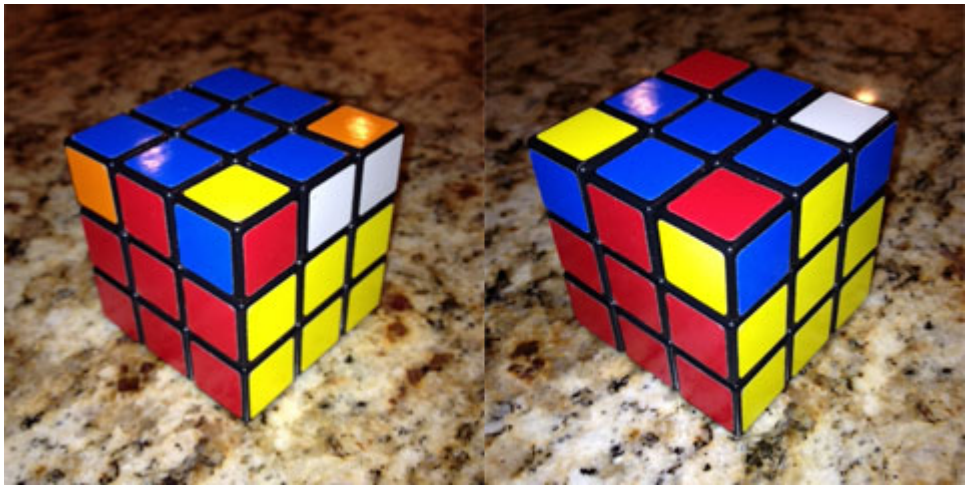
corner cubelets also get rotated in the process. This is a move that could be useful in creating an alternate solution to Rubik's cube.



$FRUR^{-1}U^{-1}F^{-1}$

Now that we have the *edge facelets* on the up face properly oriented, we just need to permute the up face *edge cubelets* until they are all in their proper positions. The algorithm we use for this is $RUR^{-1}URU^2R^{-1}$. Let's break this down a bit. First, perform this algorithm, and keep track of the up-front *edge cubelet*. What you should notice is that as you do U, U, and U^2, the up-front *edge cubelet* basically just winds up right back where it started. No change. However, notice also the presence of the *conjugates* RUR^{-1} and RU^2R^{-1} in our algorithm. Basically what we are doing with these *conjugates* is that we are moving an *edge cubelet* out of the up-right position, rotating the up face, and then moving our *edge cublet* back into the up-right position, and the end result of $RUR^{-1}URU^2R^{-1}$ is that we permute three of the *edge cubelets* on the up face with one another. Also, fortunately, nothing below the top layer is disturbed once we complete our $RUR^{-1}URU^2R^{-1}$ algorithm, and the cycle structure for this algorithm is $(UL\ \ UR\ \ UB)(UBL\ \ UFR)(ULF\ \ URB)$. When I apply this algorithm, I usually begin with my red-blue *edge cubelet* in its proper position, and then I repeat the

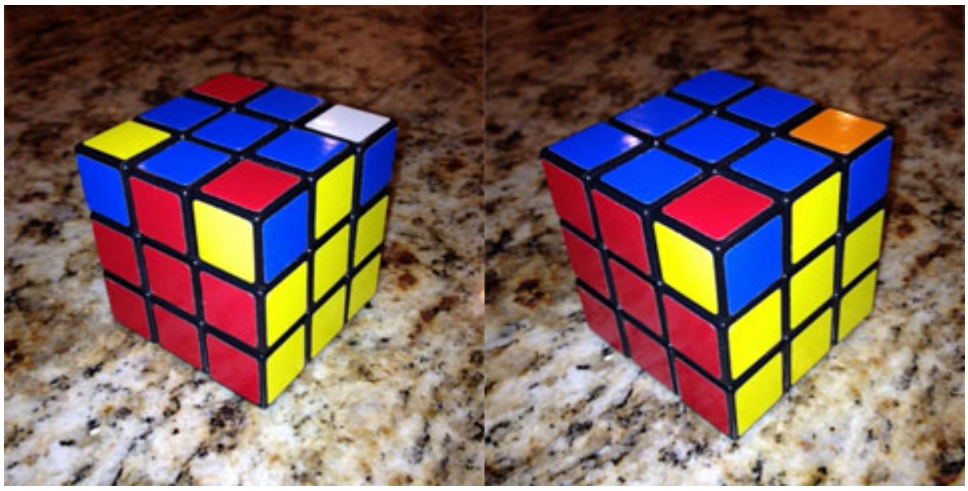
algorithm until the yellow-blue *edge cubelet* is properly placed. Then, if I need to, I rotate the whole cube so that I'm looking at the white face, I repeat the algorithm one more time from that position, and then I rotate the up face $90°$ clockwise, and I'm done. Notice, too, that if we did this algorithm three times, then the result would be $(UBL \quad UFR)(ULF \quad URB)$ which means that we are just switching, on the up face, two back *corner cubelets* diagonally with two front *corner cubelets*. Again, this, in itself, could be a useful algorithm for an alternate solution to Rubik's cube.



$RUR^{-1}URU^2R^{-1}$ done twice

At this point, we just need to permute our *corner cubelets* using the algorithm $URU^{-1}L^{-1}UR^{-1}U^{-1}L$. Embedded in this algorithm we can see the *conjugates* URU^{-1} and $(L^{-1}U)R^{-1}(U^{-1}L)$. Also, if we removed all the turns of the up face from this algorithm, then we would be left with the *commutator* $RL^{-1}R^{-1}L$, but by itself this algorithm equals the *identity* since *R* and *L* commute with one another. Hence, we need the rotations of the up face thrown in order to achieve something meaningful. And all that this algorithm wonderfully does is to permute three *corner cubelets* on the up face, $(ULF \quad URB \quad UBL)$. Usually, you want to turn your cube so that the *corner cubelet* in the up-right-front position is already properly placed. If none of the *corner cubelets* are in their correct position, then just

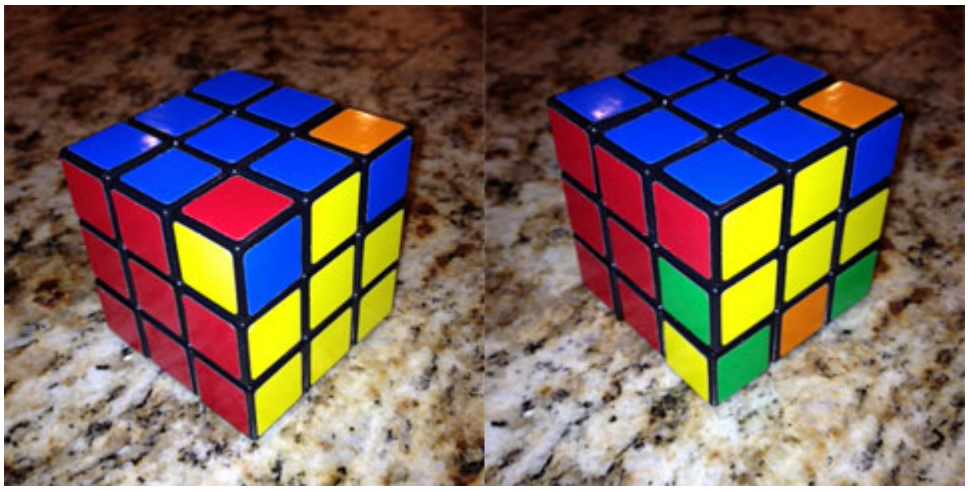
perform this algorithm once, and you should be able to find one that you can make the up-right-front *cubelet* just by turning the whole cube. And from there, just keep repeating the algorithm until all the *corner cubelets* on the up face are in their correct positions. As before, if you perform these algorithms step by step in a software program, then it's a lot easier to see what's going on!



$URU^{-1}L^{-1}UR^{-1}U^{-1}L$ done twice

Our final move is to simply rotate the *corner cubelets* on the up face until they get the proper orientation and the cube is solved. To do this, we use the algorithm $\left(R^{-1}D^{-1}RD\right)^2$. Notice that the core of this algorithm is the *commutator* $R^{-1}D^{-1}RD$, and the cycle structure for this algorithm is $(DB \quad DR \quad FR)$. The good news is that this algorithm leaves the up-right-front *cubelet* right where it is, and when we perform it, we also see that it rotates the up-right-front *cubelet* $120°$ counterclockwise which is equivalent to a clockwise rotation of $240°$. The bad news, of course, is that it messes up the rest of the cube. However, since we have a cycle of length 3, that means that if we perform the algorithm three times, then nothing is left messed up. Now here's the cool part. Remember that in our previous chapter on counting the number of permutations in Rubik's cube we saw that every turn of a face would collectively rotate our *corner cubelets* some multiple of $360°$. Well, that means that when we get down to this final point in solving the cube, our last four *corner cubelets* are collectively going to be rotated

by some multiple of $360°$, and since each application of $\left(R^{-1}D^{-1}RD\right)^2$ results in a rotation of a the up-right-front *corner cubelet* by $120°$ counterclockwise, the number of times we're going to have to perform this algorithm is going to be some multiple of three, and thus, in the end none of the rest of the cube will be disturbed. Hence, we position a *corner cubelet* that needs to be rotated in the up-right-front position and apply $\left(R^{-1}D^{-1}RD\right)^2$ until its right, and then we move another *corner cubelet* on the up face into that position and apply $\left(R^{-1}D^{-1}RD\right)^2$ again until it's correctly oriented. And the end result is that our cube is solved. And it's just that simple!



$\left(R^{-1}D^{-1}RD\right)^2$

Rotate the up face and apply $\left(R^{-1}D^{-1}RD\right)^2$ twice

Stick a fork in it, it's done!

SUMMARY (PART 8)

As promised, in part 8 we've covered:

- *Conjugates* applied to Rubik's cube.
- *Commutators* applied to Rubik's cube.
- The *commutator* or *derived subgroup.*
- The *center* of a *group*.
- The *orbit* of an element of a set acted upon by a *group*.
- Special *subgroups* of the *Rubik's cube group*.
- Additional ways to use GAP to study Rubik's cube.
- A deeper analysis of the solution to Rubik's cube.
- How to count the number of possible permutations of Rubik's cube.

PRACTICE (PART 8)

For each *group* below, use GAP to find the *center* and the *commutator* (*derived*) *subgroup*, and show the size and a list of elements of each. Also find the *commutator factor group* and show its multiplication table. And finally, find the orbits for each *group*.

1. $D_3 \cong S_3$ *with generators* $(1,2,3)$ *and* $(2,3)$.

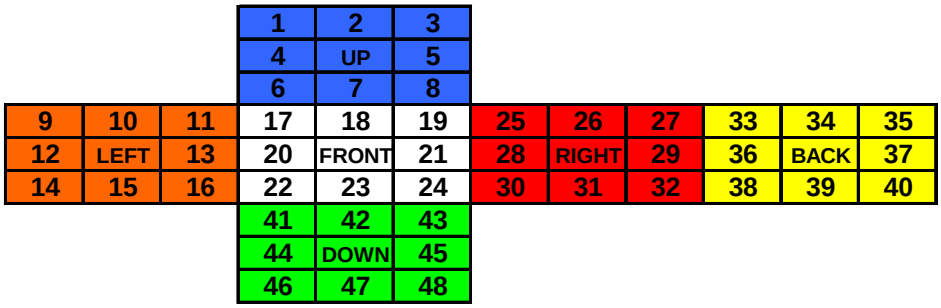
2. D_4 *with generators* $(1,2,3,4)$ *and* $(2,4)$.

3. Q_8 *with generators* $(1,2,5,6)(3,8,7,4)$ *and* $(1,4,5,8)(2,7,6,3)$.

4. A_5, *the alternating group of degree 5.*

5. *The subgroup of the Rubik's cube group generated by r and u. NOTE: Do not attempt to show the elements in this commutator (derived) subgroup. It is too large. Also, remember to first save the following into a text file called rubik.txt on your C-drive. Additionally, recall our number scheme for the facelets on Rubik's cube.*

```
r:=(25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24);
l:=(9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35);
u:=(1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19);
d:=(41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40);
f:=(17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11);
b:=(33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27);
```


1 2 3
4 UP 5
6 7 8
9 10 11 17 18 19 25 26 27 33 34 35
12 LEFT 13 20 FRONT 21 28 RIGHT 29 36 BACK 37
14 15 16 22 23 24 30 31 32 38 39 40
41 42 43
44 DOWN 45
46 47 48

PRACTICE (PART 8) - ANSWERS

For each *group* below, use GAP to find the *center* and the *commutator* (*derived*) *subgroup*, and show the size and a list of elements of each. Also find the *commutator factor group* and show its size and multiplication table and state the *abelian group* that this *quotient* (*factor*) *group* is *isomorphic* to.

1. $D_3 \cong S_3$ *with generators* $(1,2,3)$ *and* $(2,3)$.

```
gap> d3:=Group((1,2,3),(2,3));
Group([ (1,2,3), (2,3) ])

gap> Size(d3);
6

gap> c:=Center(d3);
Group(())

gap> Size(c);
1

gap> Elements(c);
[ () ]

gap> d:=DerivedSubgroup(d3);
Group([ (1,3,2) ])

gap> Size(d);
3

gap> Elements(d);
[ (), (1,2,3), (1,3,2) ]

gap> q:=CommutatorFactorGroup(d3);
Group([ f1 ])

gap> Size(q);
2

gap> ShowMultiplicationTable(q);
*                | <identity> of ... f1
-----------------+-----------------------------------
<identity> of ... | <identity> of ... f1
f1               | f1               <identity> of ...
```

Isomorphic to C_2

```
gap> o:=Orbits(d3);
[ [ 1, 2, 3 ] ]

gap> Size(o);
1
```

2. D_4 *with generators* $(1,2,3,4)$ *and* $(2,4)$.

```
gap> d4:=Group((1,2,3,4),(2,4));
Group([ (1,2,3,4), (2,4) ])

gap> Size(d4);
8

gap> c:=Center(d4);
Group([ (1,3)(2,4) ])

gap> Size(c);
2

gap> Elements(c);
[ (), (1,3)(2,4) ]

gap> d:=DerivedSubgroup(d4);
Group([ (1,3)(2,4) ])

gap> Size(d);
2

gap> Elements(d);
[ (), (1,3)(2,4) ]

gap> q:=CommutatorFactorGroup(d4);
<pc group with 2 generators>

gap> Size(q);
4

gap> ShowMultiplicationTable(q);
*                | <identity> of ... f1               f2               f1*f2
-----------------+-----------------------------------------------------------
------------
<identity> of ... | <identity> of ... f1               f2               f1*f2
f1               | f1                <identity> of ... f1*f2            f2
f2               | f2                f1*f2             <identity> of ... f1
f1*f2            | f1*f2             f2                f1
<identity> of ...
```

Isomorphic to $C_2 \times C_2$.

```
gap> o:=Orbits(d4);
[ [ 1, 2, 3, 4 ] ]

gap> Size(o);
1
```

3. Q_8 *with generators* $(1,2,5,6)(3,8,7,4)$ *and* $(1,4,5,8)(2,7,6,3)$.

```
gap> q8:=Group((1,2,5,6)*(3,8,7,4),(1,4,5,8)*(2,7,6,3));
Group([ (1,2,5,6)(3,8,7,4), (1,4,5,8)(2,7,6,3) ])

gap> Size(q8);
8
```

```
gap> c:=Center(q8);
Group([ (1,5)(2,6)(3,7)(4,8) ])

gap> Size(c);
2

gap> Elements(c);
[ (), (1,5)(2,6)(3,7)(4,8) ]

gap> d:=DerivedSubgroup(q8);
Group([ (1,5)(2,6)(3,7)(4,8) ])

gap> Size(d);
2

gap> Elements(d);
[ (), (1,5)(2,6)(3,7)(4,8) ]

gap> q:=CommutatorFactorGroup(q8);
<pc group with 2 generators>

gap> Size(q);
4

gap> ShowMultiplicationTable(q);
*                 | <identity> of ... f1                f2                f1*f2
------------------+-------------------------------------------------------------
------------
<identity> of ... | <identity> of ... f1                f2                f1*f2
f1                | f1                <identity> of ... f1*f2             f2
f2                | f2                f1*f2             <identity> of ... f1
f1*f2             | f1*f2             f2                f1
<identity> of ...
```

Isomorphic to $C_2 \times C_2$.

```
gap> o:=Orbits(q8);
[ [ 1, 2, 4, 5, 7, 3, 6, 8 ] ]

gap> Size(o);
1
```

4. A_5*, the alternating group of degree 5.*

```
gap> a5:=AlternatingGroup(5);
Alt( [ 1 .. 5 ] )

gap> Size(a5);
60

gap> Center(a5);
Group(())

gap> c:=Center(a5);
Group(())

gap> Size(c);
1

gap> Elements(c);
[ () ]
```

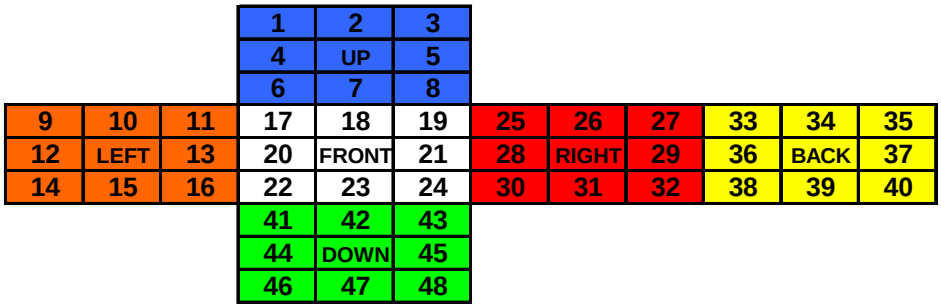
```
gap> d:=DerivedSubgroup(a5);
Alt( [ 1 .. 5 ] )

gap> Size(d);
60

gap> Elements(d);
[ (), (3,4,5), (3,5,4), (2,3)(4,5), (2,3,4), (2,3,5), (2,4,3), (2,4,5),
(2,4)(3,5), (2,5,3), (2,5,4), (2,5)(3,4),
  (1,2)(4,5), (1,2)(3,4), (1,2)(3,5), (1,2,3), (1,2,3,4,5), (1,2,3,5,4),
(1,2,4,5,3), (1,2,4), (1,2,4,3,5),
  (1,2,5,4,3), (1,2,5), (1,2,5,3,4), (1,3,2), (1,3,4,5,2), (1,3,5,4,2),
(1,3)(4,5), (1,3,4), (1,3,5), (1,3)(2,4),
  (1,3,2,4,5), (1,3,5,2,4), (1,3)(2,5), (1,3,2,5,4), (1,3,4,2,5), (1,4,5,3,2),
(1,4,2), (1,4,3,5,2), (1,4,3),
  (1,4,5), (1,4)(3,5), (1,4,5,2,3), (1,4)(2,3), (1,4,2,3,5), (1,4,2,5,3),
(1,4,3,2,5), (1,4)(2,5), (1,5,4,3,2),
  (1,5,2), (1,5,3,4,2), (1,5,3), (1,5,4), (1,5)(3,4), (1,5,4,2,3), (1,5)(2,3),
(1,5,2,3,4), (1,5,2,4,3), (1,5,3,2,4),
  (1,5)(2,4) ]

gap> q:=CommutatorFactorGroup(a5);
Group(())

gap> Size(q);
1

gap> ShowMultiplicationTable(q);
*  | ()
---+---
() | ()
```

Isomorphic to $\{(\ )\}$, the *identity*.

```
gap> o:=Orbits(a5);
[ [ 1, 2, 3, 4, 5 ] ]

gap> Size(o);
1
```

5. *The subgroup of the Rubik's cube group generated by r and u. NOTE: Do not attempt to show the elements in this commutator (derived) subgroup. It is too large. Also, remember to first save the following into a text file called rubik.txt on your C-drive. Additionally, recall our number scheme for the facelets on Rubik's cube.*

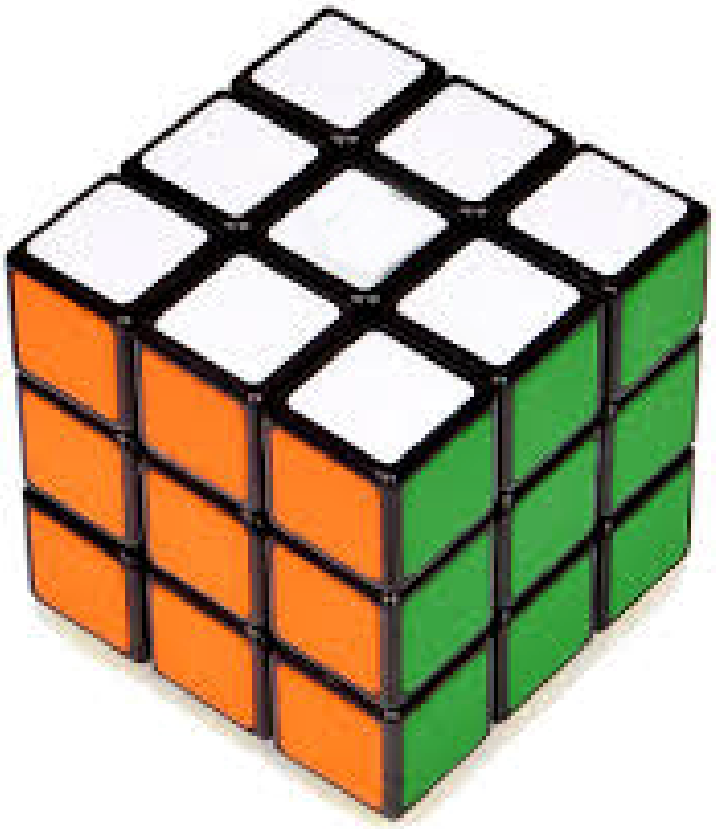
```
r:=(25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24);
l:=(9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35);
u:=(1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19);
d:=(41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40);
f:=(17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11);
b:=(33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27);
```




```
gap> Read("C:/rubik.txt");

gap> r:= (3,38,43,19)(5,36,45,21)(8,33,48,24)(25,27,32,30)(26,29,31,28);
(3,38,43,19)(5,36,45,21)(8,33,48,24)(25,27,32,30)(26,29,31,28)

gap> u:= (1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19);
(1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19)

gap> g:=Group(r,u);
<permutation group with 2 generators>

gap> Size(g);
73483200

gap> c:=Center(g);
Group([ (1,9,35)(3,33,27)(6,17,11)(8,25,19)(24,30,43)(32,38,48) ])


gap> Size(c);
3

gap> Elements(c);
[ (), (1,9,35)(3,33,27)(6,17,11)(8,25,19)(24,30,43)(32,38,48),
(1,35,9)(3,27,33)(6,11,17)(8,19,25)(24,43,30)(32,48,
    38) ]

gap> d:=DerivedSubgroup(g);
<permutation group with 3 generators>

gap> Size(d);
36741600

gap> q:=CommutatorFactorGroup(g);
<pc group with 1 generators>

gap> Size(q);
2

gap> ShowMultiplicationTable(q);
*                 | <identity> of ... f1
------------------+-----------------------------------
<identity> of ... | <identity> of ... f1
f1                | f1                <identity> of ...
```

Isomorphic to C_2

```
gap> o:=Orbits(g);
[ [ 1, 3, 38, 8, 43, 33, 6, 19, 48, 25, 11, 24, 27, 17, 35, 32, 9, 30 ], [ 2, 5,
36, 7, 45, 4, 21 ],[ 10, 34, 26, 29, 18, 31, 28 ] ]

gap> Size(o);
3
```



KNOWLEDGE OF THE UNIVERSE IS CONTAINED WITHIN THE RUBIK'S CUBE OF SPACE!