5-2-2018

Rubik's Cube: The Invisible Solve

Allen Charest

Rubik’s Cube: The Invisible Solve

Allen Charest

Submitted in Partial Completion of the
Requirements for Commonwealth Honors in Mathematics

Bridgewater State University

May 2, 2018

Dr. Ward Heilman, Thesis Advisor
Dr. Irina Seceleanu, Committee Member
Dr. Rachel Stahl, Committee Member

Table of Contents

Abstract:

The Rubik’s Cube is one of the most popular and recognizable puzzles ever made. In this research, we use group theory to identify and analyze the different solutions for the Rubik’s Cube and its variations. Since they cannot be seen on a standard Rubik’s Cube, these different solutions are called invisible solves. But by putting specialized labels on each of the center pieces of a Rubik’s Cube, we are able to track each of the invisible solves and see how they are different from one another. Dependent on the size of the Rubik’s Cube, the number of distinct invisible solves varies. For example, the 3x3x3 cube has only one center piece on each side; but the 4x4x4 has four different centers on each side. This difference in centers changes the total number of invisible solves. In addition to finding the number of invisible solves in the 3x3x3 and 4x4x4 cases, we also determine, in the 3x3x3 case, how each solve can be produced using certain algorithms. In the case of the 4x4x4 cube, we can use the parity theorem (the difference between odd and even) to verify that the number of invisible solves is correct. This research provides new insight about the structure of different solutions to the Rubik’s cube and its variations.

Introduction:

When Ernő Rubik first invented the Rubik’s Cube in 1974, he had no idea how popular it would become. Originally, he had designed the cube to see if there was a way to move all of the individual parts without the entire cube falling apart. It was not until he scrambled it and did not know how to solve it again that he realized he had actually created a puzzle. After spending a month learning how to solve the cube, Rubik decided to try and popularize his product on a worldwide scale. Although the entire process took six years, the Rubik’s Cube become a craze during the 1980s and eventually, one of the best-selling games of all time. In the early days, many players were only focused on finding the best way to solve the Rubik’s Cube from a given original layout. Since there were no instructions on how to solve the cube, the most common method was trial and error. Eventually, numerous books and articles were published which allowed people to solve the cube in under a minute. As these books and articles became more and more available to everyone, interest died out and sales ultimately fell drastically compared to their levels in the 1980s. Although interest rose again in the 2000s due to recreational activities like speed-solving competitions, most people believed there wasn’t really anything new to learn about the Rubik’s cube.

The basic design of the structure of a standard Rubik’s Cube is a 3x3x3 cube made out of 26 cubies. These cubies consist of 12 edge pieces (each of which contains two different colors), 8 corner pieces (each of which contains three different colors), and 6 center pieces (each of which contains one color). There are six different moves that can be made on a 3x3x3 cube: U^1, L^1, F^1, R^1, B^1 and D^1. These six letters correspond to quarter-turn clockwise twists about the up, left, front, right, back and down faces, respectively. In this case, the term “clockwise” refers to the direction to turn the face when you are looking directly at it. Thus, if you hold the cube looking at the front face, the move B^1 would appear to turn the back-face counter-clockwise. If there are any counter- clockwise turns, they are named by putting a -1 superscript after the movement; for example, a counter clockwise turn to the right is written as R^{-1}. Also, a face can be turned clockwise twice and this is indicated by R^2. The goal of the game is to produce a final configuration where all of the cubies on a side are the same color. Combinations of these different moves are then used to create more complex moves that rearrange specific parts of the cube and these combinations of moves are called algorithms. Information from "The Handbook of Cubik Math" by Alexander H. Frey and David Singmaster allowed us to come up with the names for these algorithms. [1] This handbook enables anyone interested in learning how to solve the cube. In all of the books, articles, and videos on Rubik's Cube, the idea seems to be that there is only one final state in which the cube is actually solved. This final state is when, on each side, all of the cubies on that side are the same color. In the following, we show how by labeling the center pieces, there are several distinct final states depending on how the cube is solved.

The 3x3x3 Case:

To understand how these different final states depend on how the cube is solved, we need to study the three types of pieces found on a standard Rubik's Cube: the corner piece, the side piece, and the center piece. Since both the corner and side pieces contain more than one color (the corner has three and the side has two), they cannot change their physical positions without the cube becoming unsolved. The center piece has only one color on one side; but it still ends up being in one physical place. The assumption at this point would be that each of the center pieces have only one fixed position since they cannot move their physical place. But upon closer inspection, it turns out that there is more than one way each of the center pieces can be oriented.

Although the center piece cannot move from its central place, it can rotate four different ways within its physical center location. Each center can rotate either 0º, 90º, 180º, and 270º. Once it gets to 360º, it becomes the equivalent of the center piece rotating 0º. This group of rotations can be described as $\mathbb{Z}^4$, the cyclic group with four elements corresponding to the four different rotations of a center piece. The difference in the position of the center piece for each of these rotations is shown in Figure 1.

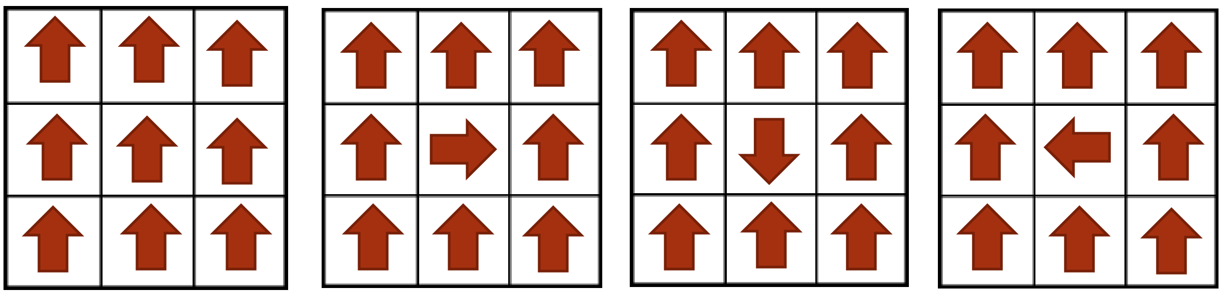
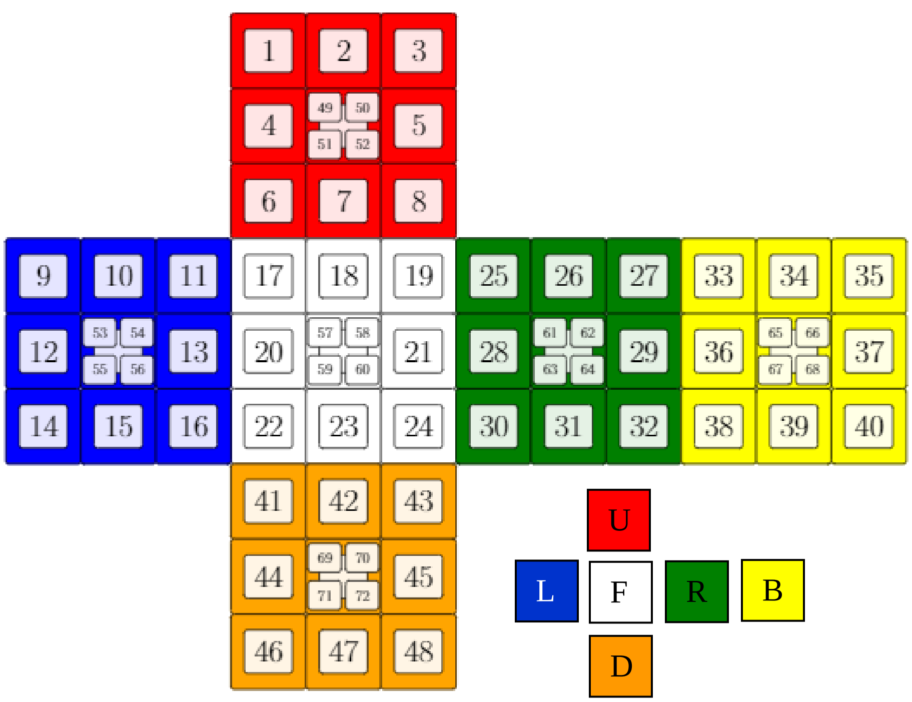


Figure #1: The group of $\mathbb{Z}^4$.

The problem is that these different rotations of the final state are not noticeable unless one has some type of label on the centers of the cube. On a standard cube, there are no labels allowing one to discern different elements of the rotational group. With that in consideration, the name we gave these different rotations is **invisible solves**. Each different rotation of any of the six center pieces creates a different and unique invisible solve for the entre cube. Ultimately, all of the invisible solves together created a group which we called the **Invisible Solves Group**. In the case of the 3x3x3 cube, the Invisible Solves Group is based off of the group $\mathbb{Z}^4$, the cyclic group with four elements corresponding to the four different rotations of a center piece. Once this group was discovered, the next step was to figure out the size of that group. For this part, there were two different approaches that we took in order to have confirmation of any results found. The two approaches used were (1) running a computer program and (2) figuring out the generators.

The computer program we used was based on a SageMath program that allows the user to look at all possible movements on the Rubik's Cube. For my research, I modified the program so that it could also track how many invisible solves are possible. Normally, the program works by giving each individual cubie a number which acts as the coding for each of the cubies. With any movement done on the cube, the numbers associated with that movement end up moving from one numbered place to another. Using these combinations of numbers, the program is able to compute all of the possible movements. When we first looked at the program, each center piece had only one number and we were not able to take all of the rotational movements of the centers into consideration. To have it recognize the invisible solves, we gave each center piece four different numbers which acted as the coding for the rotational movements of the center pieces. The result of these modifications can be seen in Figure 2 which shows both the visuals and the coding for the program in terms of the elements in the group.



U= [(9,33,25,17),(10,34,26,18),(11,35,27,19), (1,3,8,6),(2,5,7,4), (49,50,52,51)]
L= [(1,17,41,40),(4,20,44,37),(6,22,46,35),(9,11,16,14),(10,13,15,12),(53,54,56,55)]
F= [(6,25,43,16),(7,28,42,13),(8,30,41,11),(17,19,24,22),(18,21,23,20),(57,58,60,59)]
R= [(3,38,43,19),(5,36,45,21),(8,33,48,24),(25,27,32,30),(26,29,31,28),(61,62,64,63)]
B= [(1,14,48,27),(2,12,47,29),(3,9,46,32),(33,35,40,38),(34,37,39,36),(65,66,68,67)]
D= [(14,22,30,38),(15,23,31,39),(16,24,32,40),(41,43,48,46),(42,45,47,44),(69,70,72,71)]

Figure #2: The coding used for the 3x3x3 Cube.

To understand how the program works, we give an example of how the program is able to compute a move. To make a U^1 move on the Rubik's Cube, the face labeled 9 will move to the face labeled 33. That face (33) will move to the face labeled 25, that face will move to the face labeled 17, and then that face will go to where the face labeled 9 was originally. This creates a cycle that looks like this: (9,33,25,17). This same technique is applied to any of the pieces that are affected by the U^1. Then, the technique is applied to all of the possible movements on the cube.

Once all of the coding was figured out, the program gives all of the configurations possible on the Rubik's Cube. The number of possible combinations that the program produced was 88,580,102,706,155,225,088,000 $\approx 8.8 \times 10^{22}$; this is the number of different combinations that are possible on the cube when taking the different rotations of the center pieces into consideration. Next, we calculated the number of possible combinations without taking the centers into consideration. This number is 43,252,003,274,489,856,000 $\approx 4.3 \times 10^{19}$. By dividing these two numbers, we produce the number of invisible solves that were physically possible. The result of dividing these two numbers is 2,048 invisible solves. The next step was to see if that same number could be generated computationally.

When trying to figure out how to compute the number of invisible solves on the Rubik's Cube computationally, we first had to look at all of the different variables to take into consideration. For each center piece, there are 4 different ways in which it can end up and there are a total of 6 center pieces. With these two variables, there are 4*4*4*4*4*4 or 4^6 possible invisible solves. The problem is that this number is equal to 4,096 which was not equal to what the computer program produced. It seemed that a vital piece of information was missing when trying to compute the number of invisible solves: the **parity**.

In mathematics, the parity of something is its inclusion in either being even or odd. For example, the numbers 2 and 4 would have the same parity since they are both even. But the numbers 2 and 3 would not have the same parity since 2 is even and 3 is odd. With regards to the Rubik's Cube, each of the individual moves can be placed into either the even or odd category dependent on how many pieces are affected. Lets look back at the coding used to show the movement R^1. The coding for this move is: (3,38,43,19), (5,36,45,21), (8,33,48,24), (25,27,32,30), (26,29,31,28), (61,62,64,63). Within each of the cycles, there are a total of four different pieces that are being moved. As far as the parity is concerned, a 4 cycle is 3 transpositions which is an odd permutation. But since there are 6 of these 4 cycles, there are 18 transpositions so the entire move is an even permutation. This, in theory, would place all of the movements of the cube in the category of even permutations. Since all of the moves only contain an even number of transpositions, any combinations you try to do with these movements will always be even as well because combining even permutations will not produce any odd permutations. What this does is cut the number of ways to rotate each of the center pieces on the Rubik's Cube in half because half of the orientations are not physically possible. We take our original number of rotations, 4^6, and divide it by two. The number we end up with now is 2,048. This is an exact match to the number the SageMath program was producing.

So, now that we had both a computational number and a program that match that number, we confirm that the number of invisible solves on the 3x3x3 Rubik's Cube is 2,048. The next step is to show how each of them can be produced.

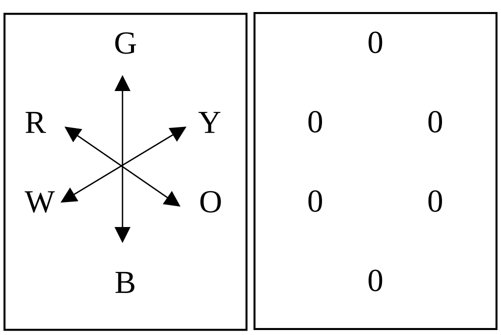
With a Rubik's Cube, there are specific algorithms that can be used to create certain patterns and move certain pieces. For the purpose of trying to create all of the invisible solves, we designed two different algorithms that allowed us to rotate the center pieces in certain ways. These algorithms are given in Figure #3:

Algorithm #1:
(R^2, U^1, R^1, U^1, R^{-1}, U^{-1}, R^{-1}, U^{-1}, R^{-1}, U^1, R^1) x 3

Algorithm #2:
(R^1, U^1, R^{-1}, U^1) x 5

Table #1: Algorithms for the 3x3x3 Rubik's Cube.

To describe how each of the center pieces move in relation to one another we display a figure that represents all of the different center pieces and where they are in relation to each other. This is how it looks:



Each of the letters represents a different colored center piece. G stands for green, R stands for red and so on for yellow, white, orange, and blue. Each of the colors is opposite the colors they would be on a normal Rubik's Cube. So in this case, green is opposite blue, red is opposite orange, and white is opposite yellow. If a specific color is not located opposite of another color, then it is located on its side. For example, the opposite color for green is blue but all of the other colors (red, yellow, white, and orange) are located on the sides of green. Each of these different-colored center pieces can be rotated either 0º, 90º, 180º, or 270º. We give each of these different rotations a specific number: 0, 1, 2, and 3 respectively to show how each of the center pieces are affected. In this specific arrangement, it shows that each of the center pieces have rotated 0º.

With each of the two algorithms listed previously, there is an associated combination of center rotations. The first algorithm allows any two centers that are side by side to each other to each be rotated 90º each. The second algorithm rotates any one center 180º while leaving the other centers unmoved. Sample diagrams for each of these are shown in Figure #3.

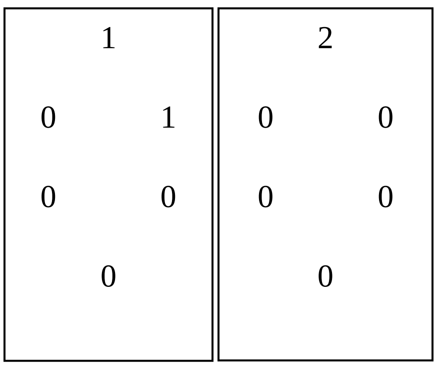


Figure #3: The image on the left shows the center rotations for Algorithm #1. The image on the right shows the center rotations for Algorithm #2.

Note that the numbers can be moved around since the only rules are that the centers have to be side by side for Algorithm #1 and only one center can be rotated for Algorithm #2. This means for Algorithm #1, there are $\binom{6}{2} - 3 = 12$ possible ways to rotate two centers side by side 90º. And for Algorithm #2, there are 6 possible ways to rotate one center 180º. We theorized that by using these algorithms in combination, we would be able to produce all 2,048 invisible solves. To do this, we considered all of the possible arrangements of center rotations and attempted to generate the invisible solves. All of the possible center rotation permutations were written as a 6-digit code and we looked at all of the possible permutations to see which ones could be produced. In relation to the previous image, the 6-digit code would start with the top number and then go around clockwise to show the rest of them. For example, the 6-digit codes for each of the two algorithms would be shown as 110000 and 200000. We were able to rule out some of the codes since it was established that only even permutations of numbers were possible because of the parity. If a code totaled an odd number, it is an odd permutation and this is not a possible invisible solve. Here is the list of 6-digit codes formed by the number of possible combinations of that code followed by the number of arrangements of that code:

000000 = 1	111122 = 15	111223 = 60	~~022223~~
111111 = 1	~~111112~~	~~112223~~	022233 = 60
222222 = 1	133333 = 6	122223 = 30	~~022333~~
333333 = 1	113333 = 15	~~122233~~	023333 = 30
~~100000~~	111333 = 20	122333 = 60	~~002333~~
110000 = 15	111133 = 15	~~123333~~	000233 = 60
~~111000~~	111113 = 6	~~112333~~	002233 = 90
111100 = 15	~~233333~~	~~111233~~	000123 = 120
~~111110~~	223333 = 15	112233 = 90	~~001123~~
200000 = 6	~~222333~~	000013 = 30	011123 = 120
220000 = 15	222233 = 15	~~000113~~	001223 = 180
222000 = 20	~~222223~~	001113 = 60	012223 = 120
222200 = 15	~~000012~~	~~011113~~	~~012233~~
222220 = 6	000112 = 60	~~011133~~	012333 = 120
~~300000~~	~~001112~~	~~011333~~	011233 = 180
330000 = 15	011112 = 30	~~013333~~	~~001233~~
~~333000~~	~~011122~~	001333 = 60	~~011223~~
333300 = 15	011222 = 60	~~000133~~	
~~333330~~	~~001222~~	001133 = 90	Total Number:
~~122222~~	~~000122~~	~~000023~~	2,048
112222 = 15	001122 = 90	~~000223~~	
~~111222~~	~~111123~~	~~002223~~	

Table #2: Possible center rotations for the 3x3x3 cube.

There are 45 sets of codes (with a total of 2,048 arrangements) that would each need an explanation as to how they are produced. In the following 45 Propositions, we show how each set of codes can be generated using Algorithms #1 and Algorithm #2. These constructions are outlined in terms of a series of propositions that are used to build other codes in subsequent propositions.

Propositions Table:

Proposition #1:

Using Algorithm #1, we can generate 12 of the possible arrangements of the code: 110000. By changing which centers we move with Algorithm #1, we can rearrange specific numbers in order to get all of the arrangements. The remaining 3 arrangements where the rotations are opposite each other on the cube will be covered in Proposition #3.

Proposition #2:

Using Algorithm #2, we can generate all of the 6 possible arrangements of the code: 200000. As before, we can use the algorithm to affect different centers in order to get all of the arrangements.

Proposition #3:

For the group 110000, there were 3 arrangements that could not be produced just from using Algorithm #1. These were the arrangements when two centers on opposite sides were rotated 90°. But by repeatedly using the properties of Propositions 1 and 2, we can produce the rest of the arrangements. As before, the numbers can be rearranged in order to produce all of the arrangements. Here is how it's done:

Step 1:

	1				0				1	
1		0	+	0		0	=	1		0
0		0		1		0		1		0
	0				1				1	

The result of each step is the beginning of the next step.

Step 2:

	1				0				1	
1		0	+	0		1	=	1		1
1		0		0		1		1		1
	1				0				1	

Step 3:

	1				0				1	
1		1	+	1		0	=	2		1
1		1		1		0		2		1
	1				0				1	

Step 4:

	1				0				1	
2		1	+	0		1	=	2		2
2		1		0		1		2		2
	1				0				1	

Step 5:

	1				0				1	
2		2	+	2		0	=	0		2
2		2		0		0		2		2
	1				0				1	

Step 6:

	1				0				1	
0		2	+	0		0	=	0		2
2		2		2		0		0		2
	1				0				1	

Step 7:

	1				0				1	
0		2	+	0		2	=	0		0
0		2		0		0		0		2
	1				0				1	

Step 8:

	1				0				1	
0		0	+	0		0	=	0		0
0		2		0		2		0		0
	1				0				1	

Proposition #4:

By repeating Proposition #2, we can produce the code: 000000. Since there are only one of these arrangements, we don't have to worry about rearranging numbers.

	2				2				0	
0		0	+	0		0	=	0		0
0		0		0		0		0		0
	0				0				0	

Proposition #5:

By using combinations from Propositions #1 and #2, we can generate this code: 333333.

Step 1:

	1				0				1	
1		0	+	0		0	=	1		0
0		0		1		0		1		0
	0				1				1	

Step 2:

	1				0				1	
1		0	+	0		1	=	1		1
1		0		0		1		1		1
	1				0				1	

Step 3:

	1				2				3	
1		1	+	0		0	=	1		1
1		1		0		0		1		1
	1				0				1	

Step 4:

	3				0				3	
1		1	+	2		0	=	3		1
1		1		0		0		1		1
	1				0				1	

Step 5:

	3				0				3	
3		1	+	0		0	=	3		1
1		1		2		0		3		1
	1				0				1	

Step 6:

	3				0				3	
3		1	+	0		0	=	3		1
3		1		0		0		3		1
	1				2				3	

Step 7:

	3				0				3	
3		1	+	0		0	=	3		1
3		1		0		2		3		3
	3				0				3	

Step 8:

	3				0				3	
3		1	+	0		2	=	3		3
3		3		0		0		3		3
	3				0				3	

Proposition #6:

By using a combination of Propositions #1 and #3, we can generate this code: 222222.

Step 1:

	1				1				2	
1		1	+	1		0	=	2		1
1		1		0		0		1		1
	1				0				1	

Step 2:

	2				0				2	
2		1	+	0		0	=	2		1
1		1		1		0		2		1
	1				1				2	

Step 3:

	2				0				2	
2		1	+	0		1	=	2		2
2		1		0		1		2		2
	2				0				2	

Proposition #7:

By using combinations from Steps 1 and 2 of Proposition #1, we can generate this code: 111111.

Step 1:

	1				0				1	
1		0	+	0		0	=	1		0
0		0		1		0		1		0
	0				1				1	

Step 2:

	1				0				1	
1		0	+	0		1	=	1		1
1		0		0		1		1		1
	1				0				1	

Proposition #8:

By using combinations from Step 1 of Proposition #1, we can generate this code: 111100. Unlike Proposition #3 where we had to do a separate Proposition to take opposite 1s into account, we do not have to do the same thing with opposite 0's since they can all be produced using the 110000 combinations from Proposition #1. As with Propositions #1 and 2, the numbers can be rearranged in order to produce all of the arrangements.

	1				0				1	
1		0	+	0		0	=	1		0
0		0		1		0		1		0
	0				1				1	

Proposition #9:

By using combinations from Proposition #2, we can generate this code: 220000. As before, the numbers can be rearranged in order to produce all of the arrangements.

	2				0				2	
0		0	+	2		0	=	2		0
0		0		0		0		0		0
	0				0				0	

Proposition #10:

By using combinations from Proposition #2 and #9, we can generate this code: 222000. As before, the numbers can be rearranged in order to produce all of the arrangements.

	2				0				2	
2		0	+	0		0	=	2		0
0		0		2		0		2		0
	0				0				0	

Proposition #11:

By using combinations from Proposition #2 and #10, we can generate this code: 222200. As before, the numbers can be rearranged in order to produce all of the arrangements.

	2				0				2	
2		0	+	0		0	=	2		0
2		0		0		0		2		0
	0				2				2	

Proposition #12:

By using combinations from Proposition #2 and #11, we can generate this code: 222220. As before, the numbers can be rearranged in order to produce all of the arrangements.

	2				0				2	
2		0	+	0		0	=	2		0
2		0		0		2		2		2
	2				0				2	

Proposition #13:

By using combinations from Proposition #1 and #9, we can generate this code: 330000. As before, the numbers can be rearranged in order to produce all of the arrangements.

	1				2				3	
1		0	+	2		0	=	3		0
0		0		0		0		0		0
	0				0				0	

Proposition #14:

By using combinations from Proposition #13, we can generate this code: 333300. As before, the numbers can be rearranged in order to produce all of the arrangements.

	3				0				3	
3		0	+	0		0	=	3		0
0		0		3		0		3		0
	0				3				3	

Proposition #15:

By using combinations from Propositions #1 and #11, we can generate this code: 112222. As before, the numbers can be rearranged in order to produce all of the arrangements.

	1				0				1	
1		0	+	0		2	=	1		2
0		0		2		2		2		2
	0				2				2	

Proposition #16:

By using combinations from Propositions #8 and #9, we can generate this code: 111122. As before, the numbers can be rearranged in order to produce all of the arrangements.

	1				0				1	
1		0	+	0		2	=	1		2
1		0		0		2		1		2
	1				0				1	

Proposition #17:

By using combinations from Propositions #2 and #5, we can generate this code: 133333. As before, the numbers can be rearranged in order to produce all of the arrangements.

	2				3				1	
0		0	+	3		3	=	3		3
0		0		3		3		3		3
	0				3				3	

Proposition #18:

By using combinations from Propositions #1 and #14, we can generate this code: 113333. As before, the numbers can be rearranged in order to produce all of the arrangements.

	1				0				1	
1		0	+	0		3	=	1		3
0		0		3		3		3		3
	0				3				3	

Proposition #19:

By using combinations from Propositions #5 and #10, we can generate this code: 111333. As before, the numbers can be rearranged in order to produce all of the arrangements.

	3				2				1	
3		3	+	2		0	=	1		3
3		3		2		0		1		3
	3				0				3	

Proposition #20:

By using combinations from Propositions #8 and #13, we can generate this code: 111133. As before, the numbers can be rearranged in order to produce all of the arrangements.

	1				0				1	
1		0	+	0		3	=	1		3
1		0		0		3		1		3
	1				0				1	

Proposition #21:

By using combinations from Propositions #2 and #7, we can generate this code: 111113. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				1				1	
0		2	+	1		1	=	1		3
0		0		1		1		1		1
	0				1				1	

Proposition #22:

By using combinations from Propositions #9 and #14, we can generate this code: 223333. As before, the numbers can be rearranged in order to produce all of the arrangements.

	2				0				2	
2		0	+	0		3	=	2		3
0		0		3		3		3		3
	0				3				3	

Proposition #23:

By using combinations from Propositions #11 and #13, we can generate this code: 222233. As before, the numbers can be rearranged in order to produce all of the arrangements.

	2				0				2	
2		0	+	0		3	=	2		3
2		0		0		3		2		3
	2				0				2	

Proposition #24:

By using combinations from Propositions #1 and #2, we can generate this code: 000112. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	0		2	=	0		2
0		1		0		0		0		1
	1				0				1	

Proposition #25:

By using combinations from Propositions #2 and #8, we can generate this code: 011112. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
1		0	+	0		2	=	1		2
1		1		0		0		1		1
	1				0				1	

Proposition #26:

By using combinations from Propositions #1 and #10, we can generate this code: 011222. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
1		0	+	0		2	=	1		2
1		0		0		2		1		2
	0				2				2	

Proposition #27:

By using combinations from Propositions #1 and #9, we can generate this code: 001122. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	0		2	=	0		2
1		0		0		2		1		2
	1				0				1	

Proposition #28:

By using combinations from Propositions #8 and #10, we can generate this code: 111223. As before, the numbers can be rearranged in order to produce all of the arrangements.

	1				0				1	
1		1	+	0		2	=	1		3
1		0		0		2		1		2
	0				2				2	

Proposition #29:

By using combinations from Propositions #1 and #28, we can generate this code: 122223. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				1				1	
1		0	+	1		3	=	2		3
1		0		1		2		2		2
	0				2				2	

Proposition #30:

By using combinations from Propositions #1 and #29, we can generate this code: 122333. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				1				1	
0		0	+	2		3	=	2		3
0		1		2		2		2		3
	1				2				3	

Proposition #31:

By using combinations from Propositions #7 and #27, we can generate this code: 112233. As before, the numbers can be rearranged in order to produce all of the arrangements.

	1				0				1	
1		1	+	0		2	=	1		3
1		1		1		2		2		3
	1				1				2	

Proposition #32:

By using combinations from Propositions #1 and #2, we can generate this code: 000013. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		1	+	0		2	=	0		3
0		1		0		0		0		1
	0				0				0	

Proposition #33:

By using combinations from Propositions #1 and #2, we can generate this code: 001113. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	0		3	=	0		3
1		0		0		1		1		1
	1				0				1	

Proposition #34:

By using combinations from Propositions #9 and #33, we can generate this code: 001333. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	0		3	=	0		3
0		2		1		1		1		3
	2				1				3	

Proposition #35:

By using combinations from Propositions #2 and #33, we can generate this code: 001133. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	0		3	=	0		3
0		2		1		1		1		3
	0				1				1	

Proposition #36:

By using combinations from Propositions #10 and #13, we can generate this code: 022233. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
2		0	+	0		3	=	2		3
2		0		0		3		2		3
	2				0				2	

Proposition #37:

By using combinations from Propositions #10 and #13, we can generate this code: 023333. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
2		0	+	0		3	=	2		3
0		0		3		3		3		3
	0				3				3	

Proposition #38:

By using combinations from Propositions #2 and #13, we can generate this code: 000233. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	0		3	=	0		3
0		0		0		3		0		3
	2				0				2	

Proposition #39:

By using combinations from Propositions #9 and #13, we can generate this code: 002233. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	0		3	=	0		3
2		0		0		3		2		3
	2				0				2	

Proposition #40:

By using combinations from Propositions #2 and #32, we can generate this code: 000123. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	0		3	=	0		3
0		2		0		0		0		2
	0				1				1	

Proposition #41:

By using combinations from Propositions #2 and #33, we can generate this code: 011123. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	1		3	=	1		3
0		2		1		0		1		2
	0				1				1	

Proposition #42:

By using combinations from Propositions #9 and #32, we can generate this code: 001223. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	0		3	=	0		3
0		2		1		0		1		2
	2				0				2	

Proposition #43:

By using combinations from Propositions #10 and #32, we can generate this code: 012223. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
0		0	+	1		3	=	1		3
2		2		0		0		2		2
	2				0				2	

Proposition #44:

By using combinations from Propositions #32 and #38, we can generate this code: 012333. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
1		3	+	0		0	=	1		3
0		0		2		3		2		3
	0				3				3	

Proposition #45:

By using combinations from Propositions #1 and #38, we can generate this code: 011233. As before, the numbers can be rearranged in order to produce all of the arrangements.

	0				0				0	
1		0	+	0		3	=	1		3
1		0		0		3		1		3
	0				2				2	

Thus, every single invisible solve of the 3x3x3 cube could be produced using just two algorithms.

Next, we considered the problem of finding how many invisible solves there are on other variations of cubes. The next largest cube and the one that was made after the 3x3x3 cube is the 4x4x4 Rubik's Revenge Cube. Since the 4x4x4 cube has some different structural designs to take into consideration, we examined it next.

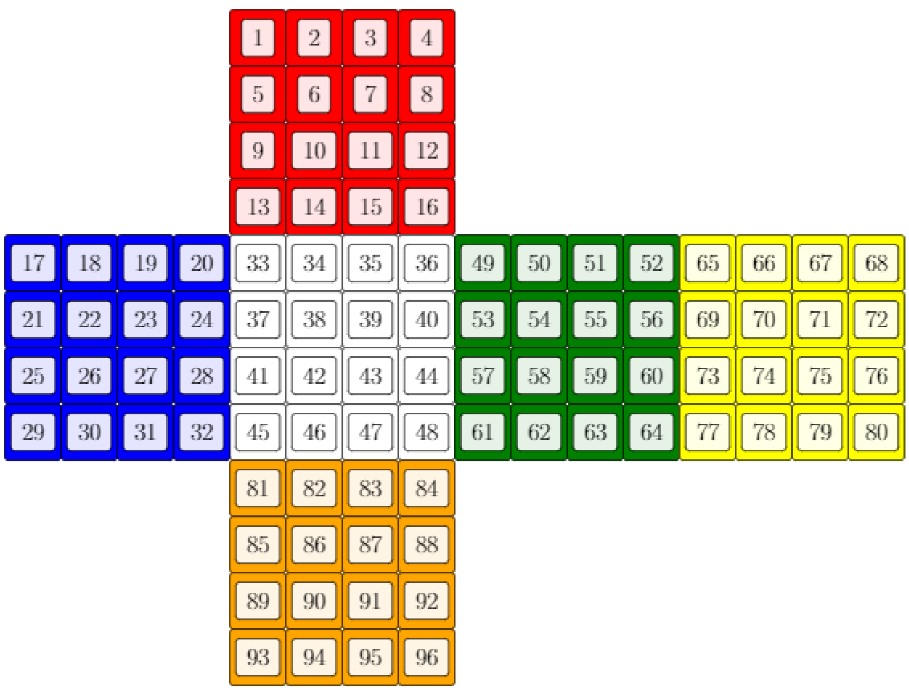
The 4x4x4 Case:

Invented in 1981 by Péter Sebestény, the Rubik's Revenge was created as a challenge to those who had already solved the 3x3x3 cube. When it first came out, as with the 3x3x3 cube, no one knew how to solve it because people thought that there would have to be completely different methods to learn. But, mathematicians eventually began to realize that there were not many differences between the 3x3x3 and 4x4x4 cube. Once these differences were identified, it became almost as easy to solve the 4x4x4 cube as it was to solve the 3x3x3 one. Even though there are not as many differences as one might expect, they significantly affect the number of invisible solves. As with the 3x3x3 cube, we implemented a SageMath program for the 4x4x4 cube first so that we would have a number to try and match with our computations. To create the program, we took all of the differences between the 3x3x3 and 4x4x4 into consideration.

The two main differences between the 3x3x3 and 4x4x4 lie in the edge pieces and the center pieces. With the 3x3x3 cube, there are 12 edge pieces that each have two different colors and none of the edge pieces have the exact same two colors. With the 4x4x4 cube, there are 24 edge pieces and each of them have two different colors. However, there are 12 pairs of two edge pieces that have the same two colors and the only way for the cube to be solved is if those two edge pieces are matched up. With the 3x3x3 cube, there is only one center on each of the six sides that can be rotated four different ways; however, the 4x4x4 cube works a bit differently. In place of one center piece, the 4x4x4 cube has four different central pieces. Instead of being able to rotate independently, these pieces move only in relation to one another. This means a single one of those 4 center pieces can only move if at least one other one moves along with it. These differences required more coding to be added to the program along with the new moves to be taken into consideration.

With the 3x3x3 cube, there are six types of moves: R^1, L^1, U^1, D^1, F^1, and B^1. But unlike the 3x3x3 cube, the structure of the 4x4x4 cube allows the centers to move their physical place creating new inner layers that have their own set of movements. To distinguish between these layers and their movements, we use different names that show when we were moving either an outer layer or an inner layer. These distinctions between layers are indicated by using the numbers 1 and 2 as subscripts. The rule of having the superscript form of the numbers represent different turns remains. The names for all of the movements on a 4x4x4 cube are R_1^1, R_2^1, L_1^1, L_2^1, U_1^1, U_2^1, D_1^1, D_2^1, F_1^1, F_2^1, B_1^1, and B_2^1. When a move has the number 1 as a subscript, it means that we are moving an outer layer and when it has the number 2 as a subscript, it means we are moving an inner layer.

All of these new moves have to be taken into consideration when writing the program to identify all moves on a 4x4x4 cube. In addition, it is likely that the centers' movements would end up affecting the total amount of moves calculated. With the 3x3x3 cube, none of the centers could move their physical place. This allowed the program to keep the cube in one position. Without having a fixed position, the program would assume you would want to take into consideration all of the different ways you can physically hold a cube. To prevent this in the 4x4x4 case, we decided to have the front upper left corner piece be fixed similar how the center piece of a 3x3x3 center piece is fixed and only put in all of the movements one could do while still keeping that corner piece fixed. In this situation, it is not possible to do the following movements: F_1^1, U_1^1, and L_1^1. All of the other moves can be done and even while taking out certain moves, we are still able to mimic them using the moves that we do have. For example, applying the sequence of moves L_2^{-1}, R_2^1, and R_1^1 is the equivalent of doing the move L_1^1. Another part to consider is that the outer and inner layers ended up having different amounts of coding cycles. When you move the R_1 layer, you end up turning one of the faces of the cube which creates 8 cycles to code. But with the R_2 layer, less pieces get moved since none of the faces turn along with it. In this situation, there are only 4 cycles to code. Once all of these factors were considered, we created a program to see how many invisible solves it could track. Here is how it looked:



R1 = [(36,4,77,84), (40,8,73,88), (44,12,69,92), (48,16,65,96), (49,52,64,61), (50,56,63,57), (51,60,62,53), (54,55,59,58)]
B1 = [(4,17,93,64), (3,21,94,60), (2,25,95,56), (1,29,96,52), (65,68,80,77), (66,72,79,73), (67,76,78,69), (70,71,75,74)]
D1= [(45,61,77,29), (46,62,78,30), (47,63,79,32), (48,64,80,32), (81,84,96,93), (82,88,95,89), (83,92,94,85), (86,87,91,90)]
R2 = [(35, 3,78,83), (39, 7,74,87), (43,11,70,91), (47,15,66,95)]

L2 = [(34,82,79, 2), (38,86,75, 6), (42,90,71,10), (46,94,67,14)]
F2 = [(9,50,88,31), (10,54,87,27), (11,58,86,23), (12,62,85,19)]
B2 = [(8,18,89,63), (7,22,90,59), (6,26,91,55), (5,30,92,51)]
U2 = [(37,21,69,53), (38,22,70,54), (39,23,71,55), (40,24,72,56)]
D2 = [(41,57,73,25), (42,58,74,26), (43,59,75,27), (44,60,76,28)]

Figure #4: The coding use for the 4x4x4 cube.

Again, we observe that each move consists of 4 or 8 4-cycles (3 transpositions) and so each move is an even transposition.

The program generated the number 95,551,488 as the number of all possible invisible solves. We then tried to replicate the number using computations. To do this, we evaluated the differences between the 3x3x3 and 4x4x4 to see how they would affect the numbers. The first difference to consider is the number of edge pieces; the 4x4x4 cube had 24 edge pieces whereas the 3x3x3 cube only had 12 edge pieces. Despite the difference in numbers, the same principle of an edge piece being bounded by more than one color still applies to all 24 edge pieces. Even if you tried to switch the two edge pieces that had the same two colors, they would end up rotating in a way that makes the cube unsolvable. So, the difference in edge pieces would make no difference in the number of invisible solves for the 4x4x4 cube. Also, the same principle of a 3x3x3 center piece having only one color still applies to the 4x4x4 center pieces. However, looking at a group of four center pieces on a 4x4x4, the question is in how many different ways could this group be rearranged? The symmetric group on 4 is the group of all permutations on a set of size four. In this case, the set of size four is the four centers on one face and the symmetric group gives all of the possible ways each of the pieces could move. Label each of the center pieces with the numbers 1, 2, 3, and 4. And now, imagine all of the possible ways these four numbers could be rearranged. The different ways they can be interchanged is shown below:

1234	1423	2314	3124	3412	4213
1243	1432	2341	3142	3421	4231
1324	2134	2413	3214	4123	4312
1342	2143	2431	3241	4132	4321

Table #3: S_4 is the 24 element symmetric group generated by transpositions.

At this point, one might assume that each group of centers can be interchanged a total of 24 different ways. This group of 24 can be described as the group S^4, the symmetric group which contains 24 elements. And like the 3x3x3 cube, there are six sides with each side containing the same number of center pieces. This calculation gives us 24*24*24*24*24 or 24^6 which is 191,102,976. But there is still one more factor to take into consideration: the parity. As with the 3x3x3 cube, the actual groups generated by the 4x4x4 cube do not allow for any odd permutations to occur. For example, in the coding for R_2, the first number, 32, moves to 3, 3 moves to 75, 75 moves to 80, and then 80 goes to where 32 originally was. This creates a set of three transpositions: (32,3,75,80) = (32,3) (32,75) (32,80). This by itself is not an even permutation. But each move of the 4x4x4 cube is made up of either 4 or 8 sets of these cycles. So that means there will be either 4*3 (12) transpositions or 8*3 (24) transpositions for any move. In either case, it is an even permutation. Since there are only even permutations, there will not be any odd permutations produced. Therefore, the original number, 191,102,976, must be divided it by 2. This gives a total of 95,551,488 possible invisible solves. This matches the number generated by the SageMath program and suggests that it is the number of invisible solves for the 4x4x4 cube. If that is the case, then with the 4x4x4 Rubik's Revenge, the Invisible Solves Group is based off of the group S_4, the symmetric group of 24 elements.

Our next step was to see if there was a way to generate every single one of these invisible solves from a minimal set of algorithms. With the 3x3x3 cube, there are fewer moves to track which makes it much easier to come up with a way to track all of them. In the 4x4x4 case however, there are too many possibilities to use the same method of finding all of the permutations. Instead, to find all of the invisible solves, a good course of action is to create algorithms that would be able to generate all of the 24 possible ways to reorient each of the center groups.

After trying out already known algorithms and making any necessary modifications, we created two algorithms that affect the centers of the 4x4x4 cube in very specific ways. The two algorithms and how they affect the center pieces are shown below:

Algorithm #1: F_2^{1}, D_2^{1}, F_2^{-1}, D_2^{-1}, U_1^{-1}, D_2^{1}, F_2^{1}, D_2^{-1}, F_2^{-1}, U_1^{1}, F_2^{-1}, D_2^{-1}, F_2^{1}, D_2^{1}, U_1^{1}, D_2^{-1}, F_2^{-1}, D_2^{1}, F_2^{-1}, U_1^{-1}, F_2^{1}, D_2^{1}, F_2^{-1}, D_2^{-1}, U_1^{-1}, D_2^{1}, F_2^{1}, D_2^{-1}, F_2^{-1}, U_1^{-1} Algorithm #2: R_2^{-1}, U_1^{1}, R_2^{-1}, D_2^{2}, R_2^{1}, U_1^{-1}, R_2^{-1}, D_2^{2}, R_2^{2}

Table #4: The two algorithms for the 4x4x4 Master Cube.

As shown in Figure #5, the first algorithm is able to switch two side-by-side center pieces with each other while rotating each of them 90° around its outer center. The second algorithm is able to switch three center pieces in a cycle with each piece rotating in relation where the original pieces were.

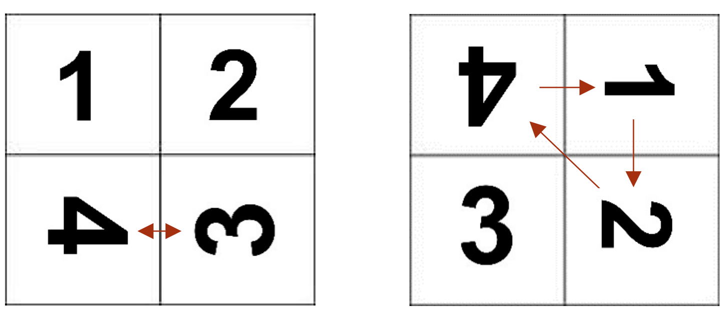


Figure #5: Movements of the center pieces that are possible by the algorithms produced.

Before we could start combining algorithms to produce the 24 possible arrangements of the center pieces, we had to consider some rules that went along with each algorithm. The first rule is a consequence of the first algorithm. When two centers on one side of the cube switch, two centers on another non-opposite side of the cube switch as well. This occurs because of the already established parity rule which forces the cube to not have an odd permutation occur. If only one switch of two center pieces was to occur, it would be an odd permutation and is therefore impossible. Since each of these center switches is equivalent to 1 transposition, we end up with 2 transpositions. The image below shows how different variations of Algorithm #1 affect two groups of center pieces.

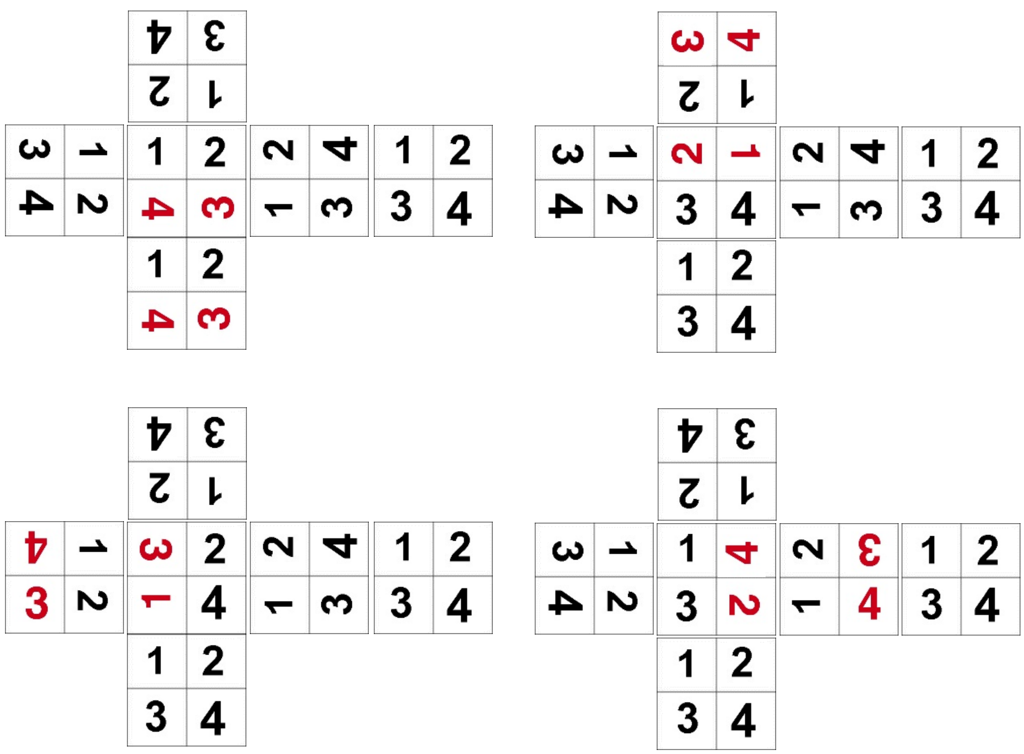


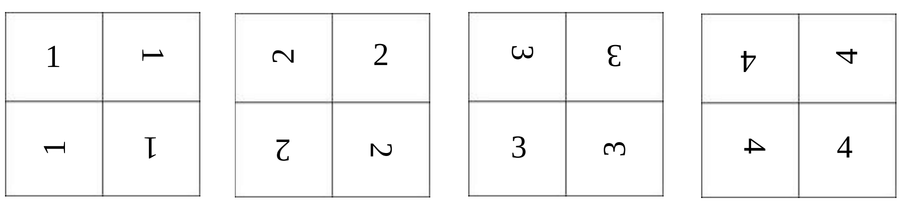
Figure #6: All of the different ways the 4x4x4 center pieces can be rearranged by Algorithm #1.

With the second algorithm, three of the center pieces on only one face are affected which creates a total of 2 transpositions. In either case, we are only having to deal with even transpositions which follows the rules set by the parity. The other rule we identified is the way centers are affected. Dependent on which way the cube is held changes drastically the way the cube is reoriented. For example, holding the cube one way and using Algorithm #1 will cause two center pieces to switch. But, if held in a different way and using the same algorithm, two horizontally different center pieces are switched instead. This rule actually plays in our favor since it gives us more options when trying to figure out how to create all of the possible arrangements of the center pieces.

Now we are able to create each of the center permutations for the 4x4x4 cube. Here is how each of them are produced:

4x4x4 Center Permutations:

With the discussion about different centers rotating, one question that needs to be asked is if each the center pieces can move similarly with the 3x3x3 center piece. From all the movements and algorithms we have done, the rotations of each of the 4x4x4 center pieces have only occurred in these ways:



But as of right now, we cannot say for certain if any of the center pieces can move in any other ways. We hope to answer that question at some point in the future.

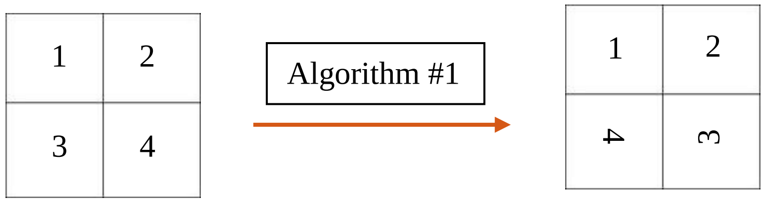
We read the permutation as expressing the order of the center cubies starting in the upper left corner, going to the upper right, then the lower left, and finally the lower right corner.

(1234):

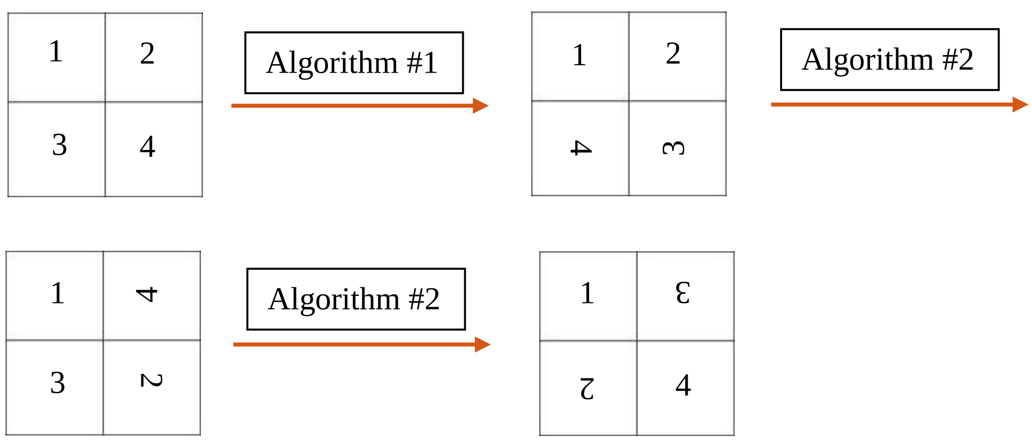
1	2
3	4

No changes needed.

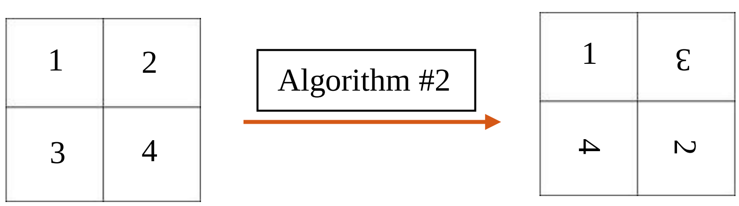
(1243):



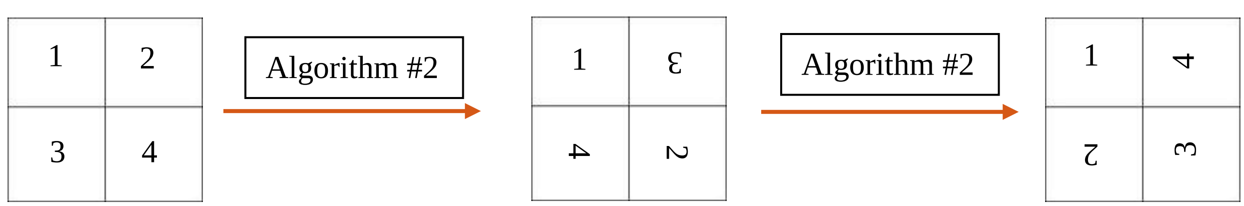
(1324):



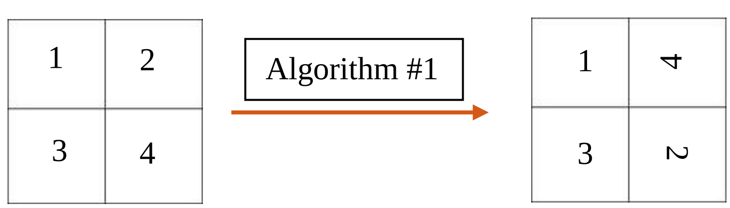
(1342):



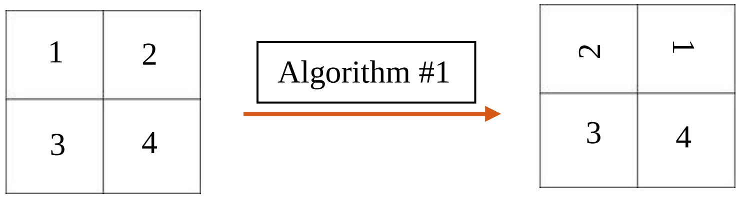
(1423):



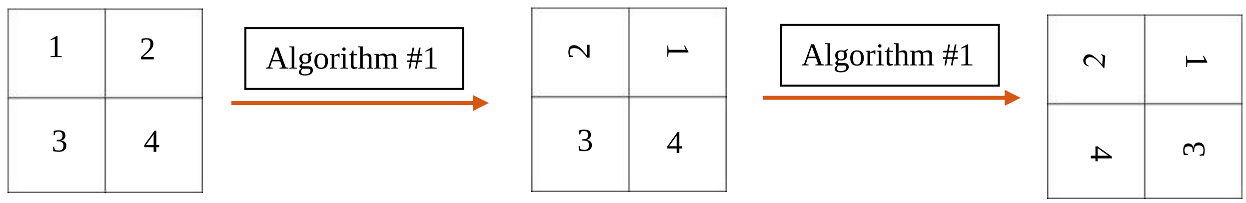
(1432):



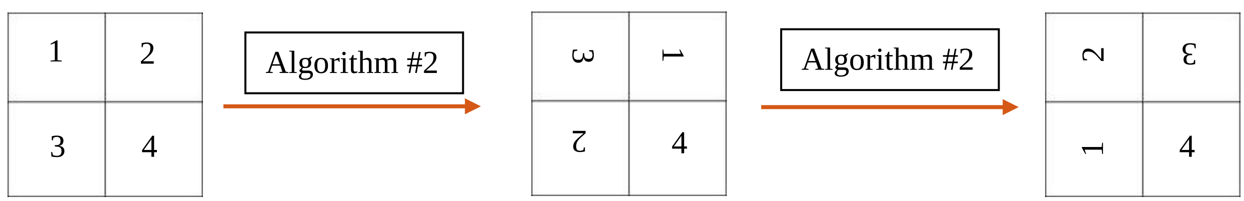
(2134):



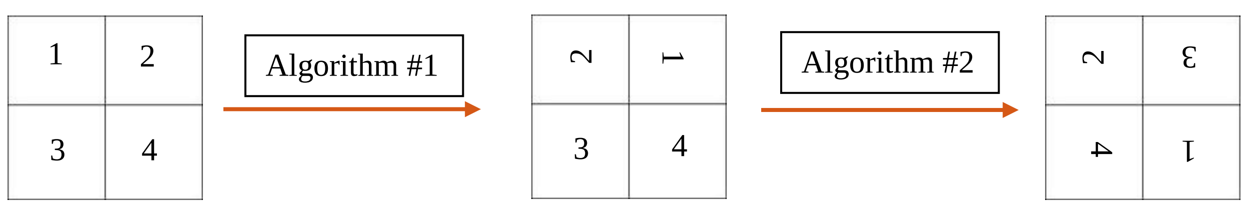
(2143):



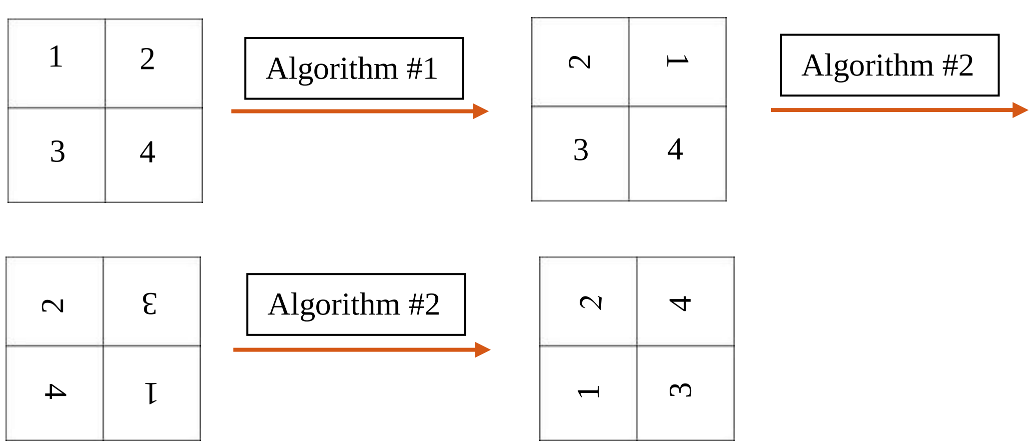
(2314):



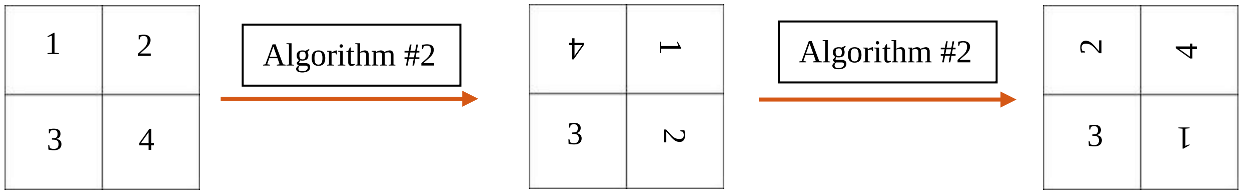
(2341):



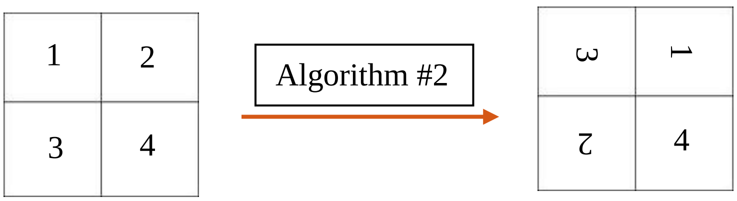
(2413):



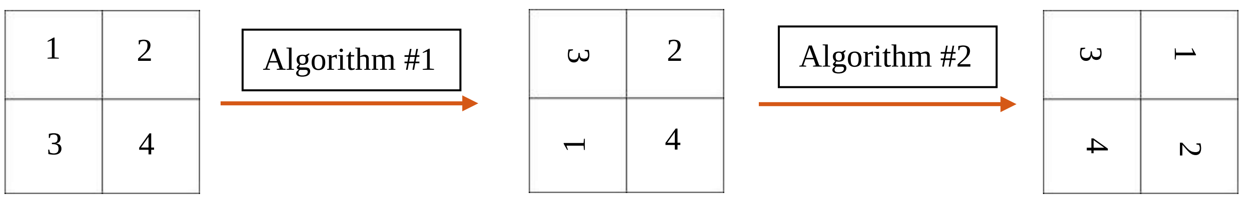
(2431):



(3124):

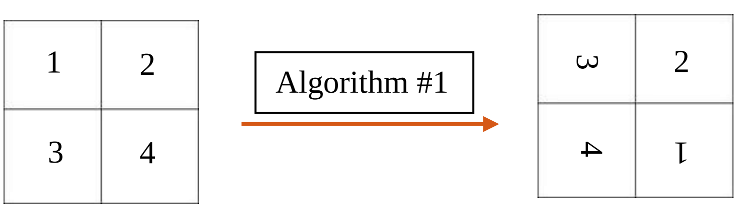


(3142):

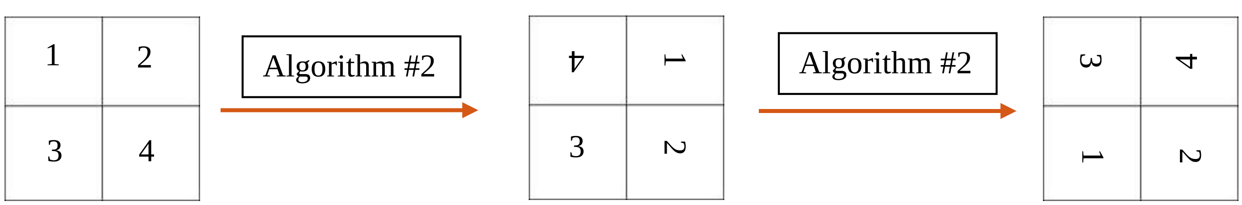


(3214):

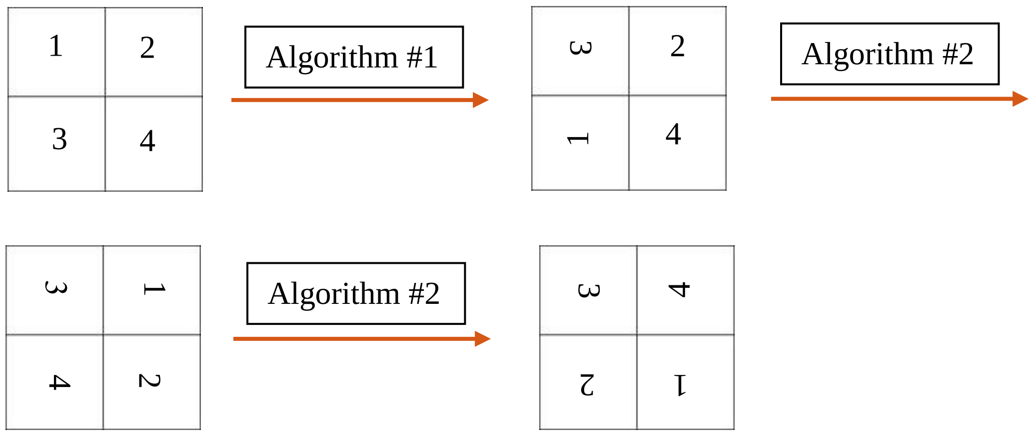
(3241):



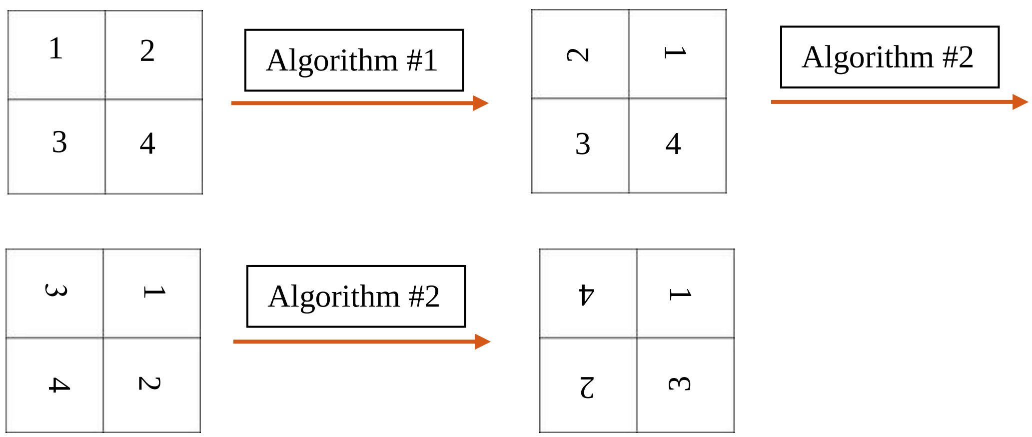
(3412):



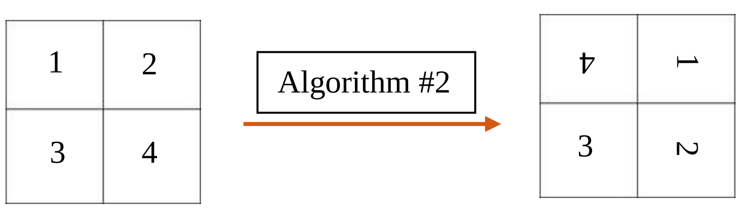
(3421):



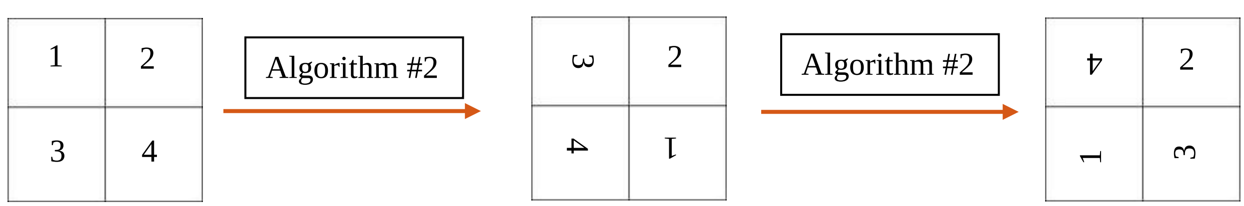
(4123):



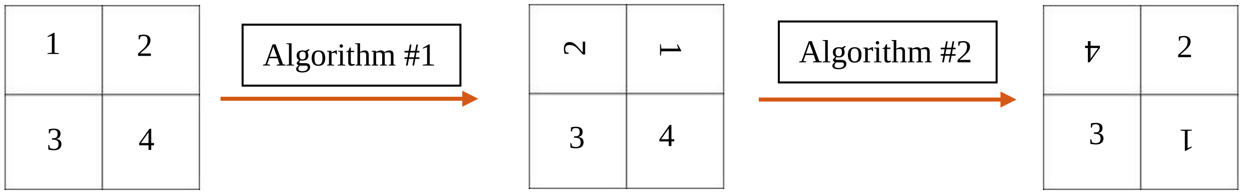
(4132):



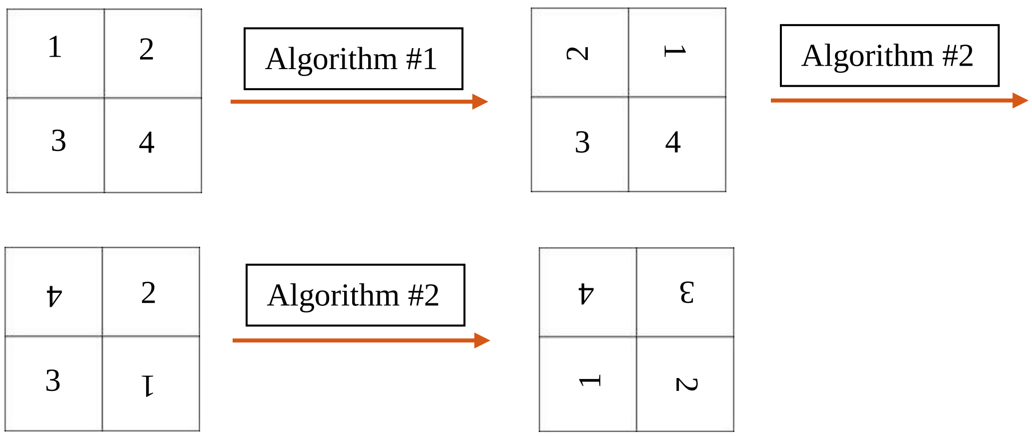
(4213):



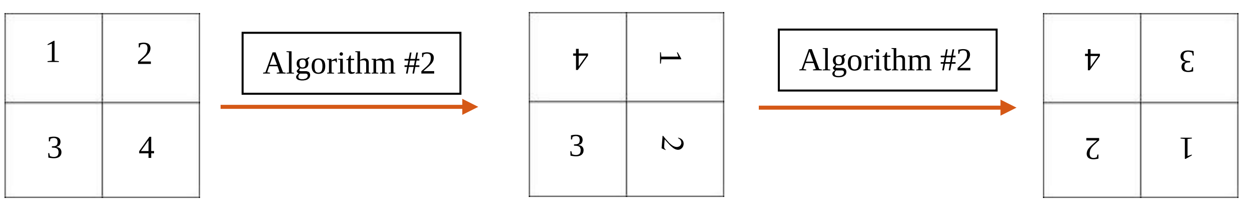
(4231):



(4312):



(4321):

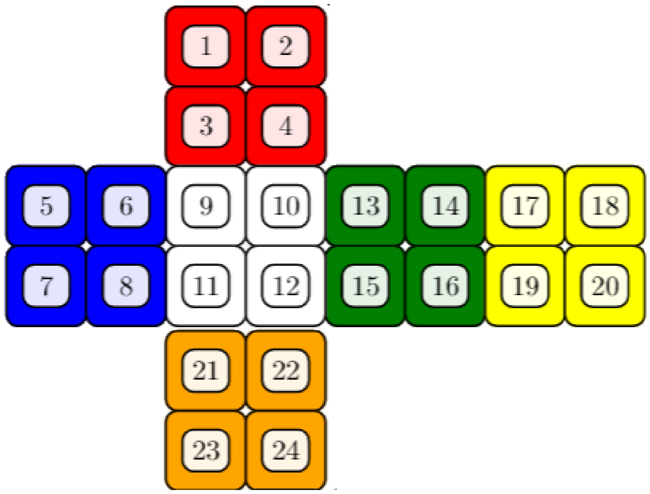


Now that we proved all of the possible arrangements of four center pieces are possible, we can show how in combination, there would be $24^6/2$ possible ways to reorient the center since we already proved that half of these orientations are not possible because of the parity. Since this is the same number we had gotten both from the program and the computations of the center pieces, we have identified and know how to generate all of the invisible solves for the 4x4x4 Rubik's Revenge. We wanted to see if we could keep doing this same method with any cube.

Our next goal was to see if the same method can be applied to any size cube. So, we decided to take a step back and tackle the 2x2x2 Pocket Cube.

The 2x2x2 Case:

Invented in 1970 by Larry D. Nichols, the 2x2x2 cube is essentially a 3x3x3 cube but without any of the center or edge pieces. Since it is made up of only corner pieces, we were curious to see if there are any possible invisible solves. Up until now, all of the invisible solves on all of the other cubes have been caused by their center pieces. So, by taking those center pieces out of the equation, the assumption would be that the 2x2x2 cube would have no invisible solves. To find out for sure, we used a similar method of using the SageMath program in order to figure out all of the combinations possible on a 2x2x2 cube. As with the 4x4x4 cube, there is no piece on the 2x2x2 cube that is in a fixed place. This affects the number the program would produce since the program would not be able to keep the cube in one fixed position. To keep the cube in one fixed position, we chose a piece on the cube that would not move and would keep the cube from being flipped around. Again, we choose the front upper left corner piece. Once this piece is kept in place, all of the single moves that would alter this piece would have to be taken out of the equation; these moves are U, L, and F. We also find the number of permutations given for a standard 2x2x2 cube in order to give the program something to compare with. The known number of permutations for a 2x2x2 cube is 3,674,160. This is the number that the program must to match if there were no invisible solves. The programming for the 2x2x2 cube is shown below:



D=[(7,11,15,19),(8,12,16,20),(21,22,24,23)]
R=[(10,2,19,22),(12,4,17,24),(13,14,16,15)]
B=[(2,5,23,16),(1,7,24,14),(17,18,20,19)]

Figure #7: The coding used for the 2x2x2 Cube.

The program generated the number 3,674,160. As mentioned before, this is the exact same number as the generally know number of permutations for the 2x2x2 cube. From this result, we concluded that the 2x2x2 cube does not have any invisible solves and that invisible solves only occur through the existence of center pieces.

Another argument we came up with is that each of the pieces on the 2x2x2 cube are composed of corner pieces that each contain 3 different colors. As was mentioned, this forces each of the faces of that corner piece to be fixed in one position; otherwise, the colors would be off and the cube would not be solved. From both the coding standpoint and a constructional standpoint, it is shown that the 2x2x2 cube only has one state in which on each side, all of the cubies on that side are the same color. Next, we move on to a cube that definitely has center pieces to look at: the 5x5x5 Professor's Cube.

The 5x5x5 Case:

Invented in 1981 by Udo Krell, the 5x5x5 cube added a new layer of difficulty both to Rubik's Cubes in general and to this particular line of research. With all of the cubes looked at previously, we only had one kind of group of invisible solves to look at. For the 3x3x3 cube, it was the cyclic group with four elements ($\mathbb{Z}^4$) and for the 4x4x4 cube, it was the symmetric group with twenty-four elements (S_4). But with the 5x5x5 cube, there are multiple groups of center pieces that have to be taken into consideration. First, consider the set of 6 center pieces (one on each face) that have only one physical place but multiple positions. Similarly, with the 3x3x3 cube, there is one center piece located on each side that can be rotated four different ways. On the 5x5x5 cube however, there are 8 additional centers on each side that work differently. Four of these centers are located on the corners of the group of center pieces and the other four were located in between pairs of the corner center pieces. At this point, we theorized that these two groups of center pieces behave independently from one another. This means that a corner center piece would not be able to move to where a middle or side center piece would normally go. If these groups work independently from one another, they would each have their own group of invisible solves affecting the number of possible combinations for the entire 5x5x5 cube.

The image below shows all of the center pieces on one side of the 5x5x5 cube and where they can move in relation to one another.

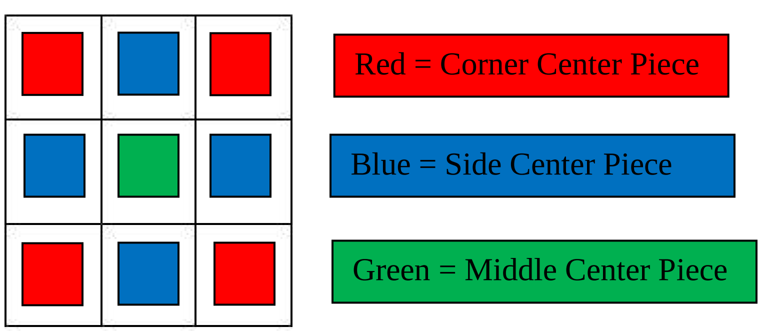
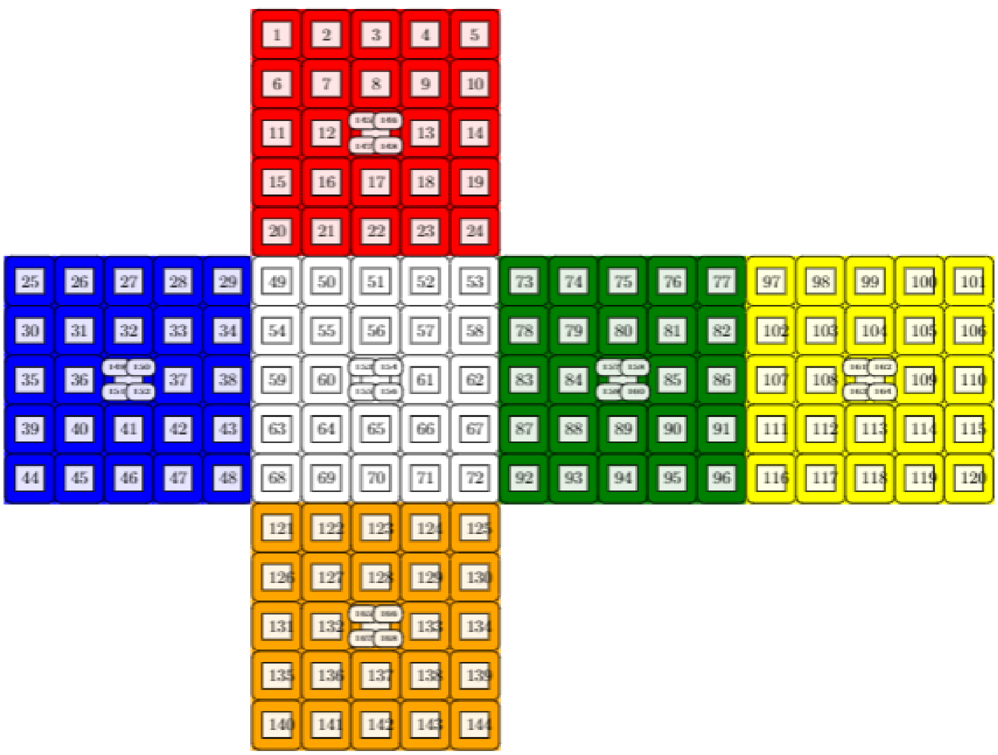


Figure #8: The different movement placed on each of the center pieces of the 5x5x5 cube.

First, look at the middle center piece. Since there are six of them and each one can be rotated four different ways, we theorized that there were $\mathbb{Z}^4$ possible movements similar with what was seen in the 3x3x3 cube. With both the corner center pieces and the side center pieces, there are four of them on each face and we theorized that each of them moves in relation only to one another of the same type similar to the 4x4x4 cube. If this hypothesis is true, then there would be two groups of S_4 elements that would be included in the amount of invisible solves To get a better idea of what this number could be, we created a program using all of the information we had available about the 5x5x5 cube and its movements.

One of the main differences between the 4x4x4 cube and the 5x5x5 center pieces is that 4x4x4 cube had no fixed center which meant we had to create one for the program to work. Whereas with the 5x5x5 cube, there is a fixed center which would keep the cube in one position in the eyes of the program. Another difference is that there are more four-cycle groups to input into the program. Before, it was noted how each of those four-cycles added up to three transpositions and that an even number of these cycles would create an even permutation. But with the 5x5x5 cube, this is not the case for some of the moves. Consider the difference between the R_1 and R_2 movements for the 5x5x5 cube. With the R_1 movement, there are a total of 12 four cycle groups each containing 3 transpositions. Of these four cycle groups, 5 of them occur on the side of the face that's being moved, 4 on the outer layer of the face, 2 on the inner layer of the face, and 1 on the center pieces of the face. This means there is a total of 36 transpositions creating an even permutation for the R_1 movement. However, with the R_2 movement, there are only 5 four cycle groups each containing 3 transpositions. This means there are only 15 transpositions; giving an odd amount of changes which is not consistent with what happens with all of the smaller cubes. The movements of both the 3x3x3 and 4x4x4 cubes create only even permutations which make it possible to prove that half of the moves were not possible because of parity. But in the 5x5x5 situation, it is not possible to prove this using the same theory since some of the moves are odd and others are even. Currently, we have not found an explanation as to how these different move sets would affect the number of invisible solves. To try and shed some light on what the number could be, we ran the program to see what it would produce. Here is an image to show how the program was done.



R1 = [(53,5,116,125), (58,10,111,130), (62,14,107,134), (67,19,102,139), (72,24,97,144), (73,77,96,92), (74,82,95,87), (75,86,94,83), (76,91,93,78), (79,81,90,88), (80,85,89,84), (157,158,160,159)]

R2 = [(52,4,117,124),(57,9,112,129),(61,13,108,133),(66,18,103,138),(71,23,98,143)]

L1 = [(49,121,120,1), (54,126,115,6), (59,131,110,11), (63,135,106,15), (68,140,101,20), (25,29,48,44), (26,34,47,39), (27,38,46,35), (28,43,45,30), (31,33,42,40), (32,37,41,36), (149,150,152,151)]

L2 = [(50,122,119,2), (55,127,114,7), (60,132,109,12), (64,136,105,16), (69,141,100,21)]
B1 = [(5,25,140,96), (4,30,141,91), (3,35,142,86), (2,39,143,82), (1,44,144,77),
(97,101,120,116), (98,106,119,111), (99,110,118,107), (100,115,117,102),
(103,105,114,112), (104,109,113,108), (161,162,164,163)]
B2 = [(10,26,135,95),(9,31,136,90),(8,36,137,85),(7,40,138,81),(6,45,139,76)]
F1 = [(20,73,125,48), (21,78,124,43), (22,83,123,38), (23,87,122,34), (24,92,121,29),
(49,53,72,68), (50,58,71,63), (51,62,70,59), (52,67,69,54), (55,57,66,64), (56,61,65,60),
(153,154,156,155)]
F2 = [(15,74,130,47), (16,79,129,42), (17,84,128,37), (18,88,127,33), (19,93,126,28)]
U1 = [(49,25,97,73), (50,26,98,74), (51,27,99,75), (52,28,100,76), (53,29,101,77), (1,5,24,20),
(2,10,23,15), (3,14,22,11), (4,19,21,6), (7,9,18,16), (8,13,17,12), (145,146,148,147)]
U2 = [(54,30,102,78),(55,31,103,79),(56,32,104,80),(57,33,105,81),(58,34,106,82)]
D1 = [(68,92,116,44), (69,93,117,45), (70,94,118,46), (71,95,119,47), (72,96,120,48),
(121,125,144,140), (122,130,143,135), (123,134,142,131), (124,139,141,126),
(127,129,138,136), (128,133,137,132), (165,166,168,167)]
D2 = [(63,87,111,39),(64,88,112,40),(65,89,113,41),(66,90,114,42),(67,91,115,43)]

Figure #9: Coding used for the 5x5x5 cube.

The program generated the number 18,698,417,887,260,966,912 ≈ 18 x 10^{18} as the number of all possible invisible solves. For each side, there are three different groups to deal with: one $\mathbb{Z}^4$cyclic group with four possible outcomes and two S^4 symmetric groups with twenty-four possible outcomes. Since there are six faces on the 5x5x5 cube, this calculation gives us $(4*24*24)^6$ possible combinations. This number is equal to 149,587,343,098,087,740,000 ≈ 1.49 x 10^{20} which ends up being exactly 8 times the number that the program calculated. One theory behind why this occurs is that for each of the three groups contained in the center pieces, the parity is reducing the amount of movements by half. Taking each group and dividing them by 2 divides the entire group by 8. This would explain why the difference between the program number and computational number is a factor of 8. The problem at this point is that there is no way to show that the groups are being affected by the parity since there are both even and odd permutations on the 5x5x5 cube. As of right now, it remains an open question how to prove the exact amount of invisible solves for this particular cube.

Conclusion:

The Rubik's Cube and its variations have been studied for over 40 years is a fascinating one. Each time we study them, there is the possibility of discovering something we may not have seen before. By looking at these hidden solves, we were able to show in the 2x2x2, 3x3x3, and 4x4x4 cases how each of them is produced and discover what they tell us about the similarity in design of these cubes. Using concepts from Group Theory and Parity, this same concept could also be applied to different shapes of cubes like pyraminxes, gear balls, and megaminxes. In the future, we may be able show there are more to these puzzles than meets the eye.

References:

[1] Alexander H. Frey, Jr. and David Singmaster. *Handbook of Cubik Math*. Cambridge: The Lutterworth Press, 1982.

[2] Joyner, David. *Adventures in Group Theory: Rubik's Cube, Merlin's Machine and Other Mathematical Toys*. Baltimore: The John Hopkins University Press, 2008.

[3] Larsen, Mogens Esrom. "Rubik's Revenge: The Group Theoretical Solution." *The American Mathematical Monthly* (1980): 381-390.

[4] Rose, John S. *A Course on Group Theory*. Mineola, New York: Dover Publications, Inc., 1978.