A 3-Cycle Guide to 3x3x3 Blindfold Cubing
Version 2.44140625

Shotaro Makisumi

January 2, 2009

Contents

Notes: Typeset in LaTeX 2_ε. The nth version is version $\left(1+\frac{1}{2^{n-1}}\right)^{2^{n-1}}$. The pictures on this guide are screenshots of Josef Jelinek's cube applet and use the Japanese color scheme. Click on the link provided with each picture to see the applet in your internet browser. HTML version of this guide also allows you to change the color scheme.

Send comments or questions to smakisumi@gmail.com.

Solving a Rubik's Cube blindfolded is much easier than you think. Cubers with only a very basic sighted method (say one minute) have learned it in less than a week, and one person has learned the method having never solved a cube with his eyes open. Unlike an advanced speedcubing method such as the Fridrich Method, even blindfolded methods used by the world's best require little memorization and rely on a few basic concepts. All you need to master blindfold cubing are an average memory and determination.

I learned 3OP (3-cycle Orientation Permutation), originally called the 3-cycle method, from Olly's Cube Page in the winter of 2002. As one of the first blindfold cubers to compete officially, I set multiple world records in 2004 and 2005 and placed second at the World Championship in 2007. Until 2007, 3OP was the method of choice among the world's fastest blindfold cubers. Although that role is now claimed by M2/R2 and the various freestyle methods, the basic principles of cycles and set-up moves remain essential in these more sophisticated methods, and 3OP can be readily applied to many other twisty puzzles.

This guide provides a detailed explanation of 3OP as used by many top blindfold cubers through 2006, including myself. While the entire method can be learned in as fast as a few days, it is enough for times as fast as 1 minute and 30 seconds, including memorization, when mastered.

Acknowledgements
This guide has been around since 2005. Special thanks are due to Sunil Pedapudi for his encouragement; to Stefan Pochmann and Richard Carr for their valuable comments on early drafts; and to Leyan Lo and Lucas Garron for their algorithms. My hope is that this guide will be of some use to new blindfold cubers.

1 Introduction

1.1 Preliminaries

The Rules of the Game
In blindfold cubing, the solver first inspects the puzzle to memorize it, without making any moves, before solving it without any aid of vision. The solver can do this by wearing an actual blindfold, as done in official competitions, by blocking the vision with a desk, by solving behind the back, or simply by closing eir eyes. In normal blindfold solving, both memorization and resolution are timed. The entire sequence is as follows:

1. Timer starts; at the same time, solver starts inspecting the puzzle. (Memorization phase)
2. Solver blocks his vision.
3. Solver solves. (Resolution phase)
4. Solver signals that he has finished solving by stopping the timer.
5. Solver unblocks his vision; if the puzzle is indeed solved, the attempt is a success.

There is also what is known as "speed blindfold cubing," in which only the resolution is timed. 3OP is designed for the first type of blindfold cubing.

Classification of Methods

The most basic classification of the various blindfold solving methods is by the way in which permutation is solved: piece-by-piece, or by decomposition into cycles. Richard Carr's piece-by-piece method belongs in first category, while all modern methods, including 3OP, use cycles. The concept of cycles will be explained later in the guide. Of the various cycle methods, the group that includes 3OP solves the orientation (the flip) before the permutation (the location), while the other group, which consists of Pochmann, M2/R2, and the various freestyle and restricted freestyle methods, combine orientation and permutation.

Whatever the method, blindfold solving a significantly different approach from speedcubing. While sighted methods aim for fewer moves and affect many pieces at each step, blindfolded methods use a limited number of basic algorithms that move very few pieces. This makes it possible to keep track of the current state of the puzzle while blindfolded. In cycle methods, the memorization encodes not the current state of the puzzle but rather the operations yet to be performed. As a result, the amount memorized becomes less and less until it becomes nothing, at which time the solver stops the timer.

1.2 Preparation

Table 1: Number Assignment (for this guide)

Number	Corners	Edges
1	UFL	UF
2	UFR	UL
3	UBR	UB
4	UBL	UR
5	DFL	FL
6	DFR	BL
7	DBR	BR
8	DBL	FR
9		DF
10		DL
11		DB
12		DR

First, pick an orientation of the cube (which top color and which front color) that you are comfortable with. Every scrambled cube will be memorized by first placing it into this orientation. In the table to the left, corners have been labeled 1 through 8 and edges 1 through 12; this labeling will be used throughout this guide for ease of discussion. In practice, you may use other numbering schemes, letters, or images, or you may rely on pure visual memory. Should you choose to use a labeling scheme, associate each label with the location and colors of that cubie. Go through each piece of a scrambled cube, numbering or labeling it appropriately and pointing to where it belongs, until you can do this without hesitation.

2 Orientation

Orientation of a cubie is its flip or twist. For each piece, we pre-define a "correct orientation." Our first goal in this method is to correct the orientation of every piece without disturbing the permutation (i.e. flip the pieces in place). Unlike in permutation, edge and corners are completely independent for orientation. We can therefore choose to start with either edges or corners.

2.1 Edge Orientation

There are twelve edge pieces on a Rubik's Cube. Since each edge has two stickers, it can be twisted in two ways: correct and incorrect. We define "correct" orientation of an edge to be the one that it can reach from the solved state within the (UDF2B2RL) group, i.e. without quarter turns on F and B faces.[1] The other orientations are "incorrect." From this definition, we can determine the orientation of an edge by (mentally) moving it to its correct position under this restriction. If the facelet colors match with the centers, the edge is correctly oriented. In official attempts we cannot make any moves during memorization.

Here is one way to process this information quickly:

In U/D layer:

1. If the piece has a U/D color, correct if this is on U/D, incorrect if on F/B/R/L.

2. Otherwise, correct if R/L color is on F/B/R/L, incorrect if on U/D.

In the middle layer:

3. If the piece has a U/D color, correct if this is on F/B, incorrect if on R/L.

4. Otherwise, look at either one of the two stickers and the adjacent center. If these two colors are same or on opposite sides, correct. Otherwise, incorrect.

Although these rules can be stated more concisely, this best approximates the way many cubers actually go about determining the edge orientation of each piece.

[1] The first two versions of this guide used the restriction (UDFBR2L2), which most top blindfold cubers used before 2007. Either one will work as long as the set-up moves for edge permutation also follow the same restriction. Using (UDF2B2RL), however, makes the set-up moves easier. Since many of the edge permutation algorithms use only R and U, we sometimes get some cancellation with the set-up moves. Also, if you start your cycles at UF, you need to move this away to L/R slice less often.

The dark-gray stickers on the applet above represent the spots where U/D sticker of a correct edge can be located.

If a piece has no U/D color, it must have a R/L color, so either rule 2 or 4 applies. The dark-gray stickers represent the spots where R/L sticker of a correct edge can be located.

Once we know how to determine the orientation, this is the easiest step in the 3-cycle method. We memorize which edges are incorrectly oriented. It can be shown using some basic group theory that any solvable configuration of the cube has an even number of incorrectly oriented edges. We can therefore proceed by flipping two of these edges at a time, which automatically corrects the orientation, with the following:

Algorithm

Flip 1 3: M'UM'UM'U2MUMUMU2 (applet)

Conjugation

Although by itself the algorithm above can only flip edges 1 and 3, through **conjugation**, we can use it to flip any two edges. In conjugation, we start with some known sequence X—in our case, the algorithm above. To modify X, we use some **set-up moves** Y and perform YXY^{-1}, that is, Y followed by X followed by the inverse of Y.[2] We demonstrate this with an example.

Example

(1)

[2]For more information on using conjugation to solve puzzles, see Jaap's Puzzle Page.

Suppose we want to flip edges 8 and 12. We take our set-up moves, Y, to be any sequence that brings edges 8 and 12 to positions 1 and 3; for example, Y =z'RB. X=M'UM'UM'U2MUMUMU2 then flips these two edges, and the inverse of the set-up moves, Y^{-1}=B'R'z, brings the edges back to their original positions. (applet)

Since conjugation and set-up moves will be used in every step of the 3-cycle, make sure that you understand these concepts. For orientation, both edges and corners, there is no restriction on the set-up moves. As we will see, however, set-up moves for the permutation steps must satisfy some conditions to make sure that the main algorithm, X, does not disturb the already-corrected orientation.

If there are more than two incorrectly oriented edges, we need to use the algorithm above, conjugated appropriately, more than once. If your goal is simply to have a successful blindfold solve, this works perfectly fine. To improve your time, however, you need to use additional algorithms that flip more than two edges at a time:

Algorithms

1 2 3 4: (M'U)*4(MU)*4 (applet)

2 3 9 11: (M'U)*4 (applet)

1 2 3 5 8 9: (RUR'F)*5 (applet)

1 3 5 6 7 8 9 11: (DwDRwR)*3 (applet)

There is also an algorithm that flips all twelve edges (called super-flip). Because of its length, however, it is not particularly useful unless we have ten or more incorrectly oriented edges. The other algorithms, although more efficient, often require clever set-up moves.

Example

(2)

To flip edges 1, 2, 5, and 7, we can set them up to the U layer with BUF, use (M'U)*4(MU)*4, then reverse the set-up moves with F'U'B'. Alternatively, the set-up moves B'UF' allow us to use the shorter four-edge flipper, (M'U)*4.
(approach 1/approach 2)

Summary
Memorization: Memorize which edges are incorrectly oriented.
Resolution: Flip the incorrectly oriented edges in groups of even numbers using the appropriate algorithms and set-up moves.

2.2 Corner Orientation

Corner orientation is slightly trickier because there are three possible orientations for each corner: correct, clockwise (hereafter "cw"), and counter-clockwise (hereafter "ccw"). A corner is correctly oriented when its U/D-colored sticker is on U or D.


Correct


Clockwise (cw)


Counter Clockwise (ccw)

What follows is a method for corner orientation based on commutators, which requires minimal memorization. A more advanced approach using faster algorithms can be found here.

It can be proven using basic group theory that, for any solvable configuration of the cube, the sum of corner orientations (where correct=0, cw=1, ccw=2) of the eight corners is always divisible by 3. This means that we can never twist a single corner by itself; two elementary changes we can do are to twist two corners in the opposite directions (**cw/ccw pair**) and to twist three corners in the same direction (**cw-triple** or **ccw-triple**). Given a scramble, we first make as many c/cw pairs as possible. The remaining incorrectly oriented corners, if any, must all have the same direction, and by the assertion above, these can always be grouped into cw-triples or ccw-triples. Corner orientation is thus reduced to solving cw/ccw pairs and cw-triples / ccw-triples. Throughout this guide, (ab) in corner orientation means to turn a ccw and b cw, and (abc cw/ccw) means turn a, b, and c all cw/ccw.

cw/ccw pair

We first explain how to solve a cw/ccw pair when both corners are on the U layer. We introduce the **monoflip**:

"Algorithm"

Monoflip A = R'D'RDR'D'R (applet)

Monoflip A'= R'DRD'R'DR (applet)

A' is the inverse of A. Note that A rotates corner 2 ccw and leaves all other U layer pieces intact. A' has a similar effect but twists 2 cw. We use what is known as a **commutator**—any sequence of the form XYX'Y', where X' and Y' represent inverses of X and Y, respectively. In particular, we set X to be A and Y to be some number of U turns. The following examples demonstrate the effect of such a commutator:

Examples

(3)

To solve (12), do U'AUA' = U'-R'D'RDR'D'R-U-R'DRD'R'DR. U' brings the ccw corner to position 2, and A rotates this corner. U then brings the cw corner to position 2, which is rotated by A'. There is no final U turn needed since the first corner is already back in its original position. *Whatever destruction A causes to the bottom two layers is reversed by A'* so that the net effect is to rotate just two corners. (applet)

(4)

(13) can be solved as U'AU2A'U' = U'-R'D'RDR'D'R-U2-R'DRD'R'DR-U'. U' brings the ccw corner to position 2, and A rotates this corner. U2 then brings the cw corner to position 2, which is rotated by A'. The final U' brings the first corner to its original position. (applet)

By using an appropriate number of U turns to set up the corners to position 2, this approach can be used to solve any c/cw pairs on U layer. It is not necessary to always start with the ccw corner; we can rotate the cw corner first by using A' first. In practice, it is much easier to perform A and A' after tilting the cube with z'. Be sure, however, to perform z at the end of the commutator. For example, Example 1 becomes z'-L'-U'R'URU'R'U-L-U'RUR'U'RU-z.

When the cw/ccw pair is not in the U layer, we use set-up moves, just like in edge orientation, to reduce it to the case above. For example, to rotate 1 ccw and 8 cw, we can set up with B'U' and rotate 2 ccw and 1 cw: B'U'-z'-U'R'URU'R'U-L'-U'RUR'U'RU-L-z-UB. Like in edge orientation, there is no restriction on the set-up moves. Alternatively, since 1 and 8 are already in L layer, we can also solve this with a commutator for that layer: U'R'URU'R'U-L2-U'RUR'U'RU-L2.

cw-triple / ccw-triple

We now use slightly different monoflips:

"Algorithm"

Monoflip C=(R'D'RD)*2 (applet)

Monoflip C'=(R'DRD')*2 (applet)

Note the addition of the final D or D'. We rely on the fact that these monoflips have order 3, meaning that CCC or C'C'C' does nothing.

Example

(5)

(123 cww) can be solved by U'CUCUCU' = U'-(R'D'RD)*2-U-(R'D'RD)*2-U-(R'D'RD)*2-U'. U' brings corner 1 to position 2, which is rotated by C. U brings the next corner to position 2, which is rotated by C. We repeat this for the third corner, and the final U' brings the first corner back to its original position. *Because C is performed three times, there is no damage done to the bottom two layers.* (applet)

Just like cw/ccw pairs, cw-/ccw-triples involving both U and D layers are handled with set-up moves. The best strategy is often to set up the corners on L layer and use an L-layer commutator. For example, (347 ccw): y'-(U'R'UR)*2-L-(U'R'UR)*2-L-(U'R'UR)*2-L2-y. Alternatively, R' reduces this to a U-layer commutator. However, we would then need to tilt the cube with z'. As another example, (257 ccw): UB'-L2-(U'R'UR)*2-L-(U'R'UR)*2-L-(U'R'UR)*2-BU'.

Additional algorithms

Although every corner orientation can be handled quite efficiently using commutators, there are slightly faster algorithms for special cases. You may wish to learn the following algorithms once you can successfully solve the corner orientation blindfolded with the approach already described.

Algorithms

(43)(21): RUR'URU'R'URU2'R'-R2U'R'U'RURURU'R
(OLL - PLL) (applet)

(43)(12): FRUR'U'RUR'U'F'-RUR'U'RwR'URU'R'w
(2 OLL's) (applet)

(234 cw): RU-R2U'R2'U-RUR'U'-RURU-RU'R'U
(Thanks to Joel van Noort!) (applet)

Even more algorithms can be found here.

Summary
Memorization: Split the orientation into cw/ccw pair(s) and/or cw-/ccw-triple(s). Memorize each group visually using the direction the U/D stickers point to.
Resolution: Solve the cw/ccw pair(s) and/or cw-/ccw-triple(s) one at a time using conjugation and monoflip commutators. Alternatively, use one of the additional algorithms together with appropriate set-up moves. In either case, set-up moves have no restriction.

3 Permutation

Permutation is the placement of the pieces. Our goal is to move all pieces to their correct spot while preserving the orientation, which should already be solved. Like orientation, permutation is also divided into corners and edges; however, each scramble has a 50% chance of having a permutation

parity, in which case we need to transpose a pair of edges and a pair of corners simultaneously. The same principle of set-up moves apply here, but with added restrictions to preserve the orientation.

3.1 Cycle Method

In this section, "corner 1" refers the corner in spot 1, not the corner that belongs to spot 1.

The permutation method explained here is know as the cycle method and is used for the corners as well as the edges. This is the defining difference between cycle methods and the so-called piece-by-piece method. It is essential that you completely understand the material in this section; solving along cycles is the single most important concept to grasp in any cycle method, including this one.

Mathematically inclined readers will recall that every permutation can be uniquely decomposed into a product of disjoint cycles (up to order of the cycles). In a more ordinary language, we can rewrite every configuration of, say, the corners, into a series of permutations in which pieces are cycled. For example, the cycle (123) means that corner 1 belongs to spot 2, 2 to 3, and 3 to 1. This decomposition of permutation into cycles can quite easily be achieved using the following:

Cycle Decomposition Algorithm

1. *Locate the smallest number that has not been written (the first time this number is 1).*
   (a) *If such number exists, write "(" and then that number.*
   (b) *If all numbers have been written, stop.*
2. *Find the last number that was written. Determine to which spot this corner needs to be moved.*
   (a) *If the number of this spot has not been written, write it down and repeat Step 2.*
   (b) *If the number of this spot has been written, write ")" to end the cycle. Go to Step 1.*

A cycle of length one means that the piece is already in place. We may disregard such cycles altogether during memorization.

The best way to see how this works is to experiment using random scrambles. We provide one example for corner permutation.

Example

(6) Scramble (on a solved cube with your chosen orientation of the cube):
R2 F2 D' L2 B2 U' R2 B2 F2 D2 L2 D' B2 U' R' F R' L' U B D R' F D U'

Start a cycle with corner 1: (1
1 belongs to 2: (12
2 belongs to 8: (128
8 belongs to 6: (1286
6 belongs to 1, completing this cycle: (1286)
Start a new cycle with corner 3, the lowest corner not yet used: (1286)(3
3 belongs to 3, completing this cycle. We disregard this cycle: (1286)(3) or (1286)
Start a new cycle with corner 4: (1286)(4
4 belongs to 5: (1286)(45

5 belongs to 7: (1286)(457
7 belongs to 4, completing this cycle: (1286)(457).

Notice that we can start a new cycle using any corner that does not already belong in a cycle. However, always starting with the corner with the lowest possible number (or earliest in some set order if no number is used) keeps the memorization simple, and less thinking means faster times. Although you must memorize everything in your head in official attempts, writing down the information on paper is a good practice when first working with cycles. As practice, apply the Cycle Decomposition Algorithm to the edges of the same scramble; you should obtain the decomposition (1 5 8)(2 6)(4 12 11 7)(9 10).

Once we have obtained a cycle decomposition, the permutation can be solved along the cycles. The 3-Cycle method is so called because 3-cycles (cycles of length 3) are used to reduce each of the cycles that make up the permutation. This relies on the following:

Cycle Reduction Rule
A cycle of length 3 or longer, when its first 3 pieces are cycled, loses the second and the third number. (More generally, a cycle of length k or longer, when the first k pieces are cycled, loses the second through the k^{th} numbers.) In particular, cycles of length 3 are reduced to cycles of length 1, which can then be discarded from memory.

For example, applying (abc) reduces (abcde) to (ade). This analysis can be performed as the cuber solves the cube, and since numbers corresponding to solved pieces can be erased from memory, we know that our solve is complete when all the information is gone.

Given a cycle decomposition, we can thus reduce the length of each cycle 2 at a time using 3-cycles. This leaves us with 2-cycles to solve for both the corners and the edges. In addition, any pair of 2-cycles of either the corners or the edges can be solved by some **double transposition** algorithm. It can be shown with using basic group theory that, after reducing each cycle in the decomposition with 3-cycles, the number of 2-cycles left for corners and for the edges are either both even or both odd. In the first case, double transpositions involving just the corners or just the edges are enough to solve the entire cube. In the second case, after some double transpositions (if any), we will be left with one 2-cycle both of the corners and of the edges. This situation, called a **permutation parity**, occurs with 50% probability.

Example

(7) We use the scramble given in the last example.

Corners: (1 2 8 6)(4 5 7)

- (1 2 8) reduces (1 2 8 6) to (1 6), leaving (1 6)(4 5 7). Since (1 6) is a 2-cycle, we cannot reduce it any further with a 3-cycle. We move on to the next cycle.
- (4 5 7) solves (4 5 7), leaving (1 6).
- We have reduced every cycle as much possible using 3-cycles. Since there is only one 2-cycle left, we cannot use a double transposition. We have a permutation parity.

Edges: (1 5 8)(2 6)(4 12 11 7)(9 10)

- (1 5 8) solves (1 5 8), leaving (2 6)(4 12 11 7)(9 10). We move on to the next cycle.
- Since (2 6) is a 2-cycle, we cannot reduce it any further with a 3-cycle. We move on to the next cycle.
- (4 12 11) reduces (4 12 11 7) to (4 7), leaving (2 6)(4 7)(9 10). Since (4 7) is a 2-cycle, we cannot reduce it any further with a 3-cycle. We move on to the next cycle.
- Since (9 10) is a 2-cycle, we cannot reduce it any further with a 3-cycle.
- We have reduced every cycle as much possible using 3-cycles. Since there are more than one 2-cycle left, we use double transpositions. (2 6)(4 7), for example, solves (2 6) and (4 7), leaving (9 10).
- Since there is only one 2-cycle left, we cannot use a double transposition. This is consistent with our observation from solving the corners that there is a permutation parity.

Permutation parity: Finally, we solve the parity CP(1 6) EP(9 10).

Be sure that you completely understand this section. The Cycle Reduction Algorithm and the Cycle Reduction Rule allow us to reduce the task of solving the permutation into applying, in appropriate order: 1) 3-cycles (corners or edges); 2) double transpositions (corners or edges); 3) and permutation parity correction. We discuss each of these in the remaining sections.

Conjugation and Set-Up Moves for Permutation
Recall the concept of conjugation, which allowed us to handle different orientations using a single algorithm combined with appropriate set-up moves. We use this techniques repeatedly in permutation as well, both corners and edges. The key difference from orientation is that we must now place certain restrictions on the set-up moves so that the main algorithm does not disturb the already-corrected orientation. The restriction depends both on the definition of orientation and the algorithms used and differ for corners and edges.

3.2 Corner Permutation

3.2.1 Cycles of Length 3

Any single algorithm that cycles 3 corners works here. For convenience, we will use one that solves the cycle (123) and its mirror, which solves (214), both of which can be performed on U/D face without disturbing the orientation.

Algorithms

CP(123): RB'RF2R'BRF2R2 (applet)

CP(214): L'BL'F2LB'L'F2L2 (applet)

As before, write our conjugated algorithms as YXY^{-1}. We can make sure that this preserves the orientation of every corner by requiring that the set-up moves, Y, preserve the orientation. From the definition of corner orientation, we can see that all U/D-layer turns, but only half turns of the four side layers, preserve the orientation. We therefore restrict the set-up moves to the (UDF2B2R2L2) group, meaning no quarter turn of the side layers. Because the moves are defined relative to some fixed placement of the center axes, we are also not free to perform cube rotations during the set-up moves. Corresponding to the restriction on the face moves, the cube rotations allowed in the set-up moves are with the (x2 y z2) group, meaning no single x or z turn.

The entire procedure for solving a 3-cycle looks like this:

1. Use set-up moves within the (UDF2B2R2L2) group to place the three corners either all on U or all on D face.
2. Permute the corners using one of the two algorithms.
3. Reverse the set-up moves.

Example

(8)

Consider the cycle (274), which can be solved as DL2D2B2-L'BL'F2LB'L'F2L2-B2D2L2D'. The set-up moves DL2D2B2 will bring the three corners to (214). The second algorithm solves this cycle, and finally we reverse the set-up moves with B2D2L2D'. (applet)

3-cycles like this example, involving two corners across a diagonal on one side and the third corner on the other side, are the hardest to set up. Some blindfold cubers instead use some variation

of the following algorithms:

Algorithms

CP(731): (R2'DR2D'R2-U2)*2 (applet)

CP(375): (R2U'R2'UR2-D2')*2 (applet)

Still others use the **Caltech cycle**, which uses the double transposition CP(24)(37): (RB'R'B)*3:

Example

(9) The same cycle (274) is solved as U-(RB'R'B)*3-U2'-(RB'R'B)*3-U with the Caltech cycle. Note that the **lone corner**, corner 7, must be permuted to position 4. We align corner 4 with corner 7 with U and switch the two with (RB'R'B)*3. U2' then aligns corner 2 with corner 4, now in position 7, and the two are switched with another (RB'R'B)*3. The final U places corner 4 in its desired destination, position 2. Note that the transpositions (24) of the two CP(24)(37) cancel each other. (applet)

3.2.2 Cycles of Length 2

Cycles of length 2 can only be solved in pairs (double transposition). The set-up moves must again stay within the (UDF2B2R2L2) group.

Algorithms

E=CP(12)(34):
xUR'U'LURU'Rw2'U'RULU'R'Ux (applet)

X=CP(13)(24):
U2 EP(13)(24), or U2RLU2R'L'F'B'U2FB (applet)

Q=CP(24)(37): (RB'R'B)*3 (applet)

CP(34)(26): (U2'RU'R'U'RU'R')*2 (inverse of Joel van Noort's)
CP(34)(15): (U2'L'ULUL'UL)*2 (mirror)

While every double transposition can be reduced to any one of these cases with clever set-up moves, we recommend learning all of these. In particular, the last three algorithms can be used to avoid long set-up moves in otherwise difficult cases. For a systematic way of handling all double transpositions, see here.

Example

(10)

Consider (28)(36). One approach is to bring all corners to U layer with the set-up moves L2DL2DL2, which reduces the permutation to (13)(24), for the full solution L2DL2DL2-U2RLU2R'L'F'B'U2FB-L2D'L2D'L2. Alternatively, setting up with DR2 reduces this to (24)(37), leading to the solution DR2-(RB'R'B)*3-R2D'.
(applets: approach 1 / approach 2)

For a systematic way to deal with all possible double transposition of corners involving both U and D layers (including some new algorithms), see here.

3.3 Edge Permutation

For many people, edge permutation is the hardest part of the 3-cycle method because it involves twelve pieces, more than the number of corners. However, the exact same approach used for corners also applies here; we will still use 3-cycles to reduce the cycles one after another. The only difference is that the set-up moves must now stay within the (UDF2B2RL) group, meaning no F/B single turns, to preserve the orientation.[3] Slice moves allowed are M2, E2, S2, and whole cube turns allowed are x2, y, and z2. The increased freedom from the set-up moves for corners also means that we need to be more careful to remember the order of turns correctly. For example, we may be able to use either B2R' or R'B2. Making some rules for the set-up moves, such as performing, whenever possible, U/D first, R/L next, and finally F2/B2, can be helpful in avoiding errors.

3.3.1 Cycles of Length 3

As with corners, it is useful to know the 3-cycle in both directions:

Algorithms

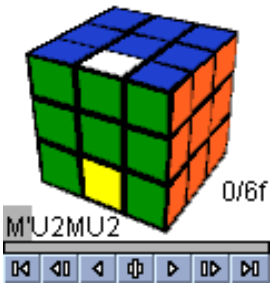
EP(421): RU'RURURU'R'U'R2 (applet)

[3]Note that this is the restriction we used to define edge orientation.

EP(241): R2URUR'U'R'U'R'UR' (applet)

These can be performed on U/D/R/L faces without disturbing the orientation.[4] Another useful 3-cycle, which, although optional, can often save a few moves, is the following and its many variations:

Algorithms



EP(193): M'U2MU2 (applet)

This can be used in any direction and on any side without disturbing the edge orientation. Of course, since every 3-cycle can be solved with either EP(421) or EP(241), we recommend that you learn to use this algorithm only after you are comfortable using the first two, and certainly not before you can complete a solve successfully.

Example

(11)

Suppose we want to do EP(156). The most basic solution is to set up the pieces on U face with L'U'L2' and use EP(412): L'U'L2'-R2URUR'U'R'U'R'UR'-L2UL More simply, recalling that these 3-cycles work on R/L faces as well, we can set up with U and perform EP(241) on L: ULz-R2URUR'U'R'U'R'UR'-z'L'U'.
(applets: approach 1 / approach 2)

The direction of every 3-cycle can be determined by just noting where one of the three pieces need to go. Therefore, while performing the set-up moves, it is enough to keep track of where the pieces go and where just one piece belongs to determine which algorithm to apply.

[4] Previous versions of this guide listed EP(243): R2U'R'U'RURURU'R and EP(423): R'UR'U'R'U'R'URUR2. Because we often start the first cycle at UF, many of our 3-cycles involve this position. This change, therefore, often eliminates a U2 from the set-up moves. For maximum efficiency, learn to use all four algorithms.

3.3.2 Cycles of Length 2

Just as with corners, 2-cycles of edges can only be solved in pairs (double transpositions). The restriction on the set-up moves are the same as for 3-cycles: (UDF2B2RL). Here, H and Z permutations are the most basic and useful algorithms.

Algorithms

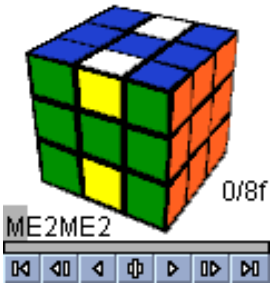
H=EP(13)(24): RLU2R'L'F'B'U2FB (applet)

Z=EP(14)(23): UR'U'RU'RURU'R'URUR2U'R'U (applet)

Z can only be used on U/D/R/L faces while H works on any face without disturbing the orientation. We also have the following useful algorithms:

Algorithms

EP(1 3)(9 11): M2U2M2U2 (applet)



EP(1 11)(3 9): ME2ME2 (applet)

EP(1 3)(7 8): (R2U2)*3 (applet)

These can be applied in any direction and on any face.

Example

(12)

Suppose we want to do EP(2 8)(6 12). The most obvious approach is to set up with RU'RL and use EP(13)(24): RU'RL-RLU2R'L'F'B'U2FB-L'R'UR'. We could also set-up with U2B2R' and use EP(14)(23) on R face: U2B2R'z'-UR'U'RU'RURU'R'URUR2U'R'U-zRB2U2. (applets: approach 1 / approach 2)

3.4 Permutation Parity

50% of solves will have a permutation parity, meaning an odd permutation of edges and, consequently, an odd permutation of the corners. Blindfold cubers have not reached a consensus on how best to deal with the permutation parity. Perhaps the easiest approach is to solve the corners and swap two edges using T permutation and then solve the four edges using H permutation. In these two parts, the set-up moves must obey the same restrictions used for corner and for edge permutation, respectively.

Algorithms

T=CP(23) EP(24): RUR'U'R'FR2U'R'U'RUR'F' (applet)

H=EP(13)(24): RLU2R'L'F'B'U2FB (applet)

Alternatively, any PLL algorithm that swaps two corners and two edges can be used together with appropriate set-up moves. In this case, the restriction on the set-up moves becomes slightly complicated. Set up the corners first within the (UDF2B2R2L2) group. Since this is more restrictive than the (UDF2B2RL) group used for the edges, these moves also preserve edge permutation. Then, set up the edges using the (UDF2B2RL) group, making sure that the two corners in question are not affected.

Example

(13)

Consider CP(17) EP(18).
Approach 1: Using the first method, we first do CP(17) EP(24) with B2U2-(T permutation)-U2B2, reducing the permutation to EP(18)(24). Now this can be handled by UR-(H permutation)-R'U'. Full solution: B2U2-RUR'U'R'FR2U'R'U'RUR'F'-U2B2-UR-RLU2R'L'F'B'U2FB-B2U2.
Approach 2: Using a longer set-up move, we can do this using one T-permutation: URU'RU'-RUR'U'R'FR2U'R'U'RUR'F'-UR'UR'U'.
Approach 3: With some insight, we can also see another nice solution: U'-(Y permutation on R)-U, or U'z'y-FRU'R'U'RUR'F'RUR'U'R'FRF'-y'zU. This works because diagonal transposition on any face does not disturb the corner orientation.
(applets: approach 1 / approach 2 / approach 3)

Permutation parity does not have to be solved at the very end. If we realize that we have parity half way into solving the permutation, we can correct the parity at an easier time. If the two pieces to be swapped are consecutive in a cycle, remember to modify this by erasing the second piece.

For many people, permutation parity is the hardest part of blindfold cubing, and it is indeed very frustrating to solve everything else successfully and make a mistake at the very last step, on

the parity. One way to avoid parities altogether is to determine the parity during inspection from the corners (corners are usually easier since there are fewer pieces). Parity is even (no parity fix necessary) if and only if the number of cycles of even length is even. If there is a parity, we can perform U at the beginning of the solve to change this (a 4-cycle is an odd permutation). However, since we cannot make any move during inspection, we must rememorize the permutation after an imaginary U.

4 Summary

We are now in a position to put everything together. The outline below walks through a blindfold solve.

Memorization
Memorization of the four parts can be done in any order.

- **Edge Permutation**
  Using the Cycle Decomposition algorithm described in Section 3.1 (Cycle Method), obtain in cycle notation the permutation of twelve edges. Memorize this.
- **Corner Permutation**
  Repeat the above for the eight corners, memorizing the cycles. Be sure to distinguish these from the permutation of edges.
- **Edge Orientation**
  Using the method explained in Section 2.1 (Edge Orientation), determine the orientation of each edge and memorize which edges are incorrectly oriented.
- **Corner Orientation**
  Memorize the direction in which the U/D sticker of each corner points.

Resolution
Orientation must be solved completely before permutation. However, within each of orientation and permutation, it does not matter whether we solve the corners or the edges first. Parity error may be corrected at any time while solving the permutation. The idea of set-up moves is crucial to understanding how we apply the algorithms. Any piece we solve can be erased from memory.

- **Corner Orientation**
  Using set-up moves and a commutator of (R'D'RD)x2 and U, solve one cw and one ccw or three in same orientation. This must usually be repeated several times to correct all orientation. We can also use conjugation and special algorithms. There is no restriction on the set-up moves.
- **Edge Orientation**
  Using set-up moves and appropriate edge orientation algorithms, flip the incorrectly oriented edges. The ones that are flipped may be erased from memory. There is no restriction on the set-up moves.

- **Corner Permutation**
  Following the Cycle Reduction Rule described in Section 3.1 (Cycle Method), apply algorithms to reduce cycles of length 3 or longer. Solve each pair of cycles of length 2 with the appropriate algorithms (double transposition). Set-up moves must be within the (UDF2B2R2L2x2yz2) group. In case a single cycle of length 2 is left, move on to edge permutation.
- **Edge Permutation**
  Repeat the same procedure for edges. Set-up moves must be within the (UDF2B2RLM2E2S2x2yz2) group.
- **Parity Fix (if necessary)**
  Use set-up moves and appropriate PLL algorithms, as described in Section 3.4 (Permutation Parity).

5 Memorization (incomplete)

For now, read this post. I visually memorize the patterns of the cycles (triangles, Z-like zigzags, parallel lines, etc).

6 Example Solves

Here are two walk-throughs of the 3-cycle method on random scrambles generated by JNetCube. Recall that (ab) in corner orientation means that a is to be turned ccw and b cw.

Examples

1. **Scramble:** D' B' F R' F2 U F L2 D2 B' U2 R2 D' L2 F R' D' F U L' F U' R B2 U'

   - **Memorization**
     CP: (1 5 4 2 7 8 3)
     EP: (1 7 9 12 11 4 5 8)(2 6)
     EO: 1 2 3 4 5 6 8 10 11 12
     CO: (2 1)(5 7 8 cw)
   - **Corner Orientation**
     (21): z'-U'R'URU'R'U-L'-U'RUR'U'RU-L-z
     (578 cw): x2z'-U'RUR'U'RUR'-L'-U'RUR'U'RUR'-L'-U'RUR'U'RUR'-L2-zx2
   - **Edge Orientation**
     1 2 3 4 9 10 11 12: x-(DwDRwR)*3-x'
     5 6 8 9: z'R-(MU)*4-R'z
   - **Edge Permutation**
     (1 7 9): U'Dz'-R'UR'U'R'U'R'URUR2-zD'U
     (1 12 11): F2x2U'-R'UR'U'R'U'R'URUR2-Ux2F2
     (1 4 5): L'U2-R2U'R'U'RURURU'R-U2L
     (1 8)(2 6): URUL-RLU2R'L'F'B'U2FB-L'U'R'U'
     No parity!

- **Corner Permutation**
  (1 5 4): D2R2U'-RB'RF2R'BRF2R2-UR2D2
  (1 2 7): DB2-RB'RF2R'BRF2R2-B2D'
  (1 8 3): D'R2D2B2-RB'RF2R'BRF2R2-B2D2R2D

2. **Scramble:** F D2 R2 D' B2 L F' B R' L U' F2 D B2 L' U2 L F' B' R' L' D2 R' L2 F'

- **Memorization**
  CP: (1 2 8 6)(4 5 7)
  EP: (1 2 5)(3 8 9 6 11 7)(4 12 10)
  EO: 1 2 4 5 6 8
  CO: (345 cw)(678 cw)
- **Corner Orientation**
  (345 cw): F2-z'(U'RUR'U'RUR'L)*3Lz-F2
  (678 cw): x2U'-z'(U'RUR'U'RUR'L)*3Lz-Ux2
- **Edge Orientation**
  1 2 4 5 6 8: BUD2-(RUR'F)*5-D2U'B'
- **Corner Permutation**
  (1 2 8): B2-RB'RF2R'BRF2R2-B2
  (4 5 7): UL2U'R2U'-RB'RF2R'BRF2R2-UR2UL2U'
  Parity left (1 6)
- **Edge Permutation**
  (1 2 5): U'L'U2-R'UR'U'R'U'RURUR2-U2LU
  (3 8 9): RF2U'-R'UR'U'R'U'RURUR2-UF2R'
  (3 6 11): U'Dz-R2U'R'U'RURURU'R-z'D'
  Parity left (3 7)
  (4 12 10): S'R2SR2
- **Parity**
  We have CP(1 6) EP(3 7). First, fix the corners and switch two additional edges:
  UR2U-(T-perm ex. RUR'U'R'FR2U'R'U'RUR'F')-U'R2U'
  This leaves us with a double transposition.
  (2 4)(3 7): U'R'-RLU2R'L'F'B'U2FB-RU

7 Links

1. **Forums**
   Blindfold Cubing subforum at Speedsolving.com
   The most active forum on blindfold cubing.

   Yahoo! Blindfold Cubing Forum
   Good place to look up past discussions on blindfold cubing. This is no longer the main forum frequented by blindfold cubers.

2. **Other documentations / techniques / algorithm lists for the 3-cycle method**
   Olly's cube page
   This is where I learned the 3-cycle method.

   An Introduction to Blindfold 3x3x3 Rubik's Cube Solving
   A guide by Tyson Mao.

   Lucas's Blindfolded Cubing Pages

   Leyan's Page
   List of algorithms. Direct link.

3. **Other methods**
   Stefan's M2/R2 blindfold cubing methods
   Very possibly the future of blindfold cubing.

   Stefan Pochmann's Blindfoldsolving
   The original Pochmann method using 2-cycles (PLL algorithms) and solving orientation and permutation simultaneously.

   Joel van Noort's Blindfold Cubing Tutorial
   A more detailed explanation of the Pochmann method.

   Richard Carr's PDF document
   Piece-by-piece method for $1 \times 1 \times 1$ up to $5 \times 5 \times 5$.

   BCFTSS (Blindfold Cubing For The Seriously Sad)
   The "intermediate" piece-by-piece method developed by Richard Carr.

   The Simplest System for Blindfold Cubing
   Orientation and permutation separate, using 2-cycles.