The Freewheeling Algorithm for solving Rubik cubes with any number of layers

Anyone writing a document like this has a target audience in mind. This document is not aimed at a beginner. This caveat allows me to shorten this document. I provide enough information even for a beginner but he will have to fill in a lot of gaps on his own.

I also do not address the speed cubist. I find no conceptual interest there.

Generalities

There seems to be consensus for the colors for the cubes. There is a white-yellow pair of opposite faces, a blue-green pair of opposite faces, and a red-orange pair of opposite faces. When held with the yellow face on the right and the white face on the left and the orange face up, the front face is blue and the back face is green and the down face is red. I refer to this as the *default* orientation.

I have seen demonstrations by solvers and instructions included with purchased cubes and they are all structured differently from the algorithm described here. The Freewheeling algorithm solves layers in symmetric pairs proceeding from the right-left outside layers toward the center. When a pair of layers is solved the layers are attached to the outer layers. The solved layers are never rotated after being solved. In effect we regard the solved layers as a single layer and refer to them collectively as the *fixed set.*

Terminology and Notation

Since the outer fixed layers are not moved there is no need for notation referring to them. So we can use the simple symbol R to refer to the first layer inside the right fixed layers and L to refer to the first layer inside the left fixed layers. These two layers are the *active* layers, where we direct our actions. The remaining layers inside the active layers are unstructured and serve as a *pool*, and need no labels.

We refer to the blocks in the center of an edge as *center edge blocks.* We refer to the edge blocks that are not at the corners or centers as *edge blocks.* We refer to the blocks on the diagonal of the faces as *diagonal blocks.* We refer to the blocks in the center layer that are not at the center nor at an edge as *center face blocks.* And we refer to the blocks that are not on an edge, a diagonal, nor the center layer as *face blocks.*

U denotes the up face, F the front face, and B the back face. We also abandon the use of the terms clockwise and counterclockwise. When we hold the cube we see three faces, one of which is unchanging. We use what we see to denote the direction of rotation. When we look at the layer R we see one of its columns in the front face. If we rotate R so that the blocks that we see move up we denote the move by Ru. If we rotate R so that these blocks move down we denote the move by Rd. If we rotate the up face so that the visible front edge moves to the right we denote the move by Ur. If the front edge moves left we denote the move by Ul. R2 and U2 are 180 degree rotations. **(U2 Ru)6** denotes a procedure in which the up layer is rotated 180 degrees, the R layer rorates up, and these two rotations are performed six times. Every procedure has a symmetric mate that is obtained by exchanging the R and L symbols.

We refer to two types of moves. A *procedure* is a fixed predetermined sequence of rotations. An example is described in the previous paragraph. They are specified in physical terms without reference to color. In a *herding* move a block, or group of blocks, is moved to a specified location, the exact route not determined ahead of time. Colors play a role in herding moves. The term used is a metaphor using the image of a shepherd who moves sheep from pasture to their stalls. A *roll* is a reorientation of the cube with no layer rotated.

I have seen videos in which a procedure is referred to as an algorithm. Algorithm is a technical mathematical term and an algorithm does not have a predetermined sequence of steps. An algorithm is a sequence of if-then-else statements. In each step there are several expected possibilities and the algorithm specifies responses to these possibilities. Look up algorithm in Wikipedia. The entire process described in this document is an algorithm, one algorithm.

The front face is divided by its diagonals into four triangles. The upper and lower triangles include the diagonal blocks and the left and right triangles do not. The upper and lower triangles are filled by procedures and the left and right triangles are filled by herding moves. The column of a layer is thus divided into three parts by its diagonal blocks.

Though we can see three faces all the action involves just the up face, the front face, and the not visible back face. The only actions involving the back face are 180 rotations and we do not have to see the back face to perform them.

Most of the cues and actions refer to two geometric shapes. A *rectangle* has its left side in L and its right side in R. A *pair* consists of two blocks of the same color both in L or both in R located symmetrically with respect to the horizontal center.

Many of the procedures we use are incapable of separating the blocks in a pair. And for that reason they also cannot join them. In the end they must be of the same color. That means that they must be of the same color before the procedures work on them. We must create pairs before we start the sequence of procedures that involve them.

Adjoints

A study of group theory in mathematics will introduce the idea of an adjoint. In group theory it is a fundamental concept. It seems, however, to be something that most students have difficulty with. It is pivotally useful in this algorithm. Here we will do little more than define the concept. If P is a procedure and A is any other procedure then **APA^{-1}** is an *adjoint* of P. Later, in context, we will go into the concept in detail to make it an accessible concept and show how it is used.

Phase One

In this phase we will configure the freewheeling faces, the yellow and white faces. Orient the cube with the white face up if there is an odd number of layers. If the number of layers is even any face will do as the up face. Fill the interior of the up face with white blocks. The moves are all herding moves but I have a suggestion. Assemble a column of white blocks in the front face, excluding the edge blocks. Then rotate this column into the up face. Repeat until the interior of the up face is white.

When the up face interior is white roll the cube so that the up face is yellow. If the number of layers is odd the up face of the center layer must be filled. Assemble the yellow blocks of the center layer in the front face of C. Rotate the front face 90 degrees in either direction. Execute Cd to drop the yellow center block into gap. Restore the front face and execute Cu.

Assemble another column of yellow blocks in the front face. With R denoting that column perform the procedure **Ru U2 Rd**. Repeat until the interior of the up face is yellow.

Retaining yellow as the up face use herding to position two yellow corner blocks correctly related to each other. Roll the cube to put the yellow face down. If you are right handed orient the cube so that the yellow corners you have produced are in the left down edge. The point is to keep from moving them and allow you to rotate the up and right faces freely. Use herding to produce a pair of yellow corners in the up face. Rotate the up face to put this pair in the right edge. Then rotate the right face to place this pair in the down face. When you have completed this step the corners of the yellow face should be correct.

With the yellow corners in place all the white corners are in the white face but they are probably not configured correctly. The first issue is to position them correctly relative to each other. If you find two corners that should be exchanged, rotate the up face to place these two corners in the up font edge and execute the following procedure.

SwapTwoCorners = Ru Ur Rd Ur Fu Ul Fd

It may be necessary to do this twice.

When they are positioned relatively correctly some may be *twisted*, rotated about an axis that is a radius of the cube. To determine what to do next it is not necessary to look at the twists. To determine what to do next it is sufficient to observe the locations of the white faces of the corner blocks. Count the number of corners that show white in the white face. The sum of the twists is always a full rotation and that fact allows us to use the pattern of white faces to determine what action to take.

- If no corner shows white up, two of the corners must be twisted 120 degrees, *1 twist*, and two must be twisted 240 degrees, *2 twist*. The 1 twist corners show white on the right side and the 2 twist corners show white on the left side. Don't bother to look at the twists, just where white appears. Rotate the up face to place a 2 twist block (white on the left), either one, in the up left corner of the cube and execute

TwistThreeCornersCW = Ru Ul Rd Ul Ru U2 Rd

This will produce a configuration with one white face up.

- If one corner shows white up, the other three corners are all 1 twists or all are 2 twists. If the corners show white on the left (240 degree twists), rotate the up face to place the white up corner in the up left corner of the cube and execute **TwistThreeCornersCW.**
-

  If the corners show white on the right (1 twists), rotate the up face to place the white up corner in the right back corner of the cube and execute

  TwistThreeCornersCCw = Ru U2 Rd Ur Ru Ur Rd

  In either case all corners will show white in the up face of the cube.
- If two corners show white up, one corner is a 1 twist and the other is a 2 twist. Rotate the up face to place the corner with white on the right (1twist) in the up left corner of the cube and execute **TwistThreeCornersCW.** This will produce a configuration with one white face up.
- Three corners showing white is impossible.
- If four corners show white we can go on to the next step.

The next step is to fill the blue, green, and red edges of both Freewheeling faces. Orient the cube in the default orientation. Use herding to fill the blue, green, and red edges. I assemble the blocks for an edge in the down front edge of the cube and sweep them into place together.

At this point each freewheeling face should be filled in correctly except for the edge blocks in the up (orange) face. If the number of layers is even this is the end of phase one.

If the number of layers is odd there is one more step. Use herding to fill in the center face blocks in the red and green faces. This move will have its payoff in the final phase.

Orient the cube in the default orientation.

Phase Two

At the end of phase one the Freewheeling faces become the fixed set. The two symmetric layers just inside them become the active layers, denoted by L and R. The first step is to configure the diagonal blocks in L and R into pairs. This is easily done by herding.

The next step is to place the left and right up face (orange) edge blocks. I have not been able to find a procedure to do this. There are just too many configurations to anticipate. You will have to use herding. This is neither obvious nor easy and it will take you many attempted rotations to achieve the required configuration. A good approach is to set intermediate goals – four edge blocks in L and R. After this things go smoothly.

The edge blocks in R and L have homes, some belong in L but are visiting in R and some belong in R but are visiting in L. If you find a visitor in L rotate L to place the visitor in the up front edge of L. If you find such a visitor there must also be a visitor in R. Rotate R to place this visitor in the down front edge of R and execute the following procedure.

TradeVisitors = U2 Ru U2

Repeat until there are no visitors.

Rotate R until you find two adjacent edge blocks showing the same color in the same face. Rotate R to place these two blocks in the up face and execute the following procedure.

CycleThreeEdges =

F2

(U2 Ru U2 Rd)

(roll exchanging front and up faces)

(U2 Ru U2 Rd)

U2

If you do not find two adjacent edges of the same color, execute the above procedure for any rotation of R. This will produce a configuration with two matching edges for a repeat of the procedure. When this is completed R can be rotated so that the edge blocks match the colors of the corner blocks.

When this is completed roll the cube to exchange R and L and repeat with the new R. Restore the default orientation.

Pairs

We are ready to attack the pairs. A pair is said to be *in parity* if its color matches the color of the face it is in, or matches the color of the opposite face. We refer to an out-of-parity pair as an OOPP. Our goal at this point is to end up with all pairs in parity. There are eight pairs on the diagonals, two for each color. The number of OOPPs is always even.

A common approach to puzzle solving is to be confronted with a situation and to look for a way to handle it. We reverse the process. We start with a procedure and look for a situation it can handle. The procedure we have in mind here is

CycleFivePairs = (U2 Ru)6

There is no rotation of L. That leaves five pairs that can be moved by CycleFivePairs, one in the up face of L and the other four in R. CycleFivePairs cycles these five pairs. The up right pair moves to up left, the up left pair moves to R in the back face, the back pair moves to the down face, the down pair moves to the front face and the front pair moves to the up face. CycleFivePairs changes the parity of four of the five pairs. Only the pair in the up face of R retains its parity. All the blocks enclosed by a pair travel with the pair.

This allows us to describe the configuration that CycleFivePairs can handle and produce the desired outcome.

> We require that of the four OOPPs one is in the up face of L and the other three are in the front, back, and down faces of R. This is the configuration required at the start of CycleFivePairs.

Since the moves in a procedure are predetermined the desired result will occur only if the configuration at the start will produce the desired outcome. We can see that the procedure will deal successfully only with the configuration described above as the start configuration.

But we will be faced with other configurations. We begin by dealing with a situation where four of the pairs are out of parity. With four OOPPs there are few possible configurations. The four OOPPs may be distributed over all four faces, one to a face. The four OOPPs may be distributed over three of the faces. In that case two OOPPs are in one face and the other two in the two adjacent faces. The four OOPPs may appear in just two adjacent faces, two in each face.

Here, what is desired is to convert OOPPs into in-parity pairs. That means that we want the up pair in L, and the pairs in the front, back, and down faces of R to be out of parity when the procedure is executed. This is the desired *start* configuration for CycleFivePairs.

To achieve our goal we will use adjoints of CycleFivePairs.

- Consider the case where the four OOPPs are distributed over all four faces. Roll the cube to obtain a configuration in which an out of parity pair is in the up face of L and R contains at least two OOPPs. This is possible unless all four are in one layer. If R contains three OOPPs execute CyclleFivePairs. Otherwise it is necessary to rotate a face to get three OOPPs in R. If you have to rotate the front face before executing CycleFivePairs it will be necessary to rotate it after executing CycleFivePairs to restore the front face. That is, you will perform **F2 CycleFivePairs F2.** Since F2 is its own inverse this operation is an adjoint of CycleFivePairs. All pairs will be in parity.

  If all four are in one layer, we could execute **U2 CycleFivePairs U2**. But that means we would be doing **U2 (U2 Ru)6 U2**, which is equal to **(Ru U2)6**, easily remembered in its own right.

- Next consider the case where the four OOPPs are distributed over three faces. Two will be in one face and of the other two, one will be in each of the adjacent faces. Orient the cube so that the pair of OOPPs in the same face are in the up face and at least two are in R. If three are in R execute **R2 CycleFivePairs R2.** This is another adjoint. If two OOPPs are in L and two in R orient the cube so that the two in L are in the up and front faces. Execute **R2 F2 CycleFivePairs F2 R2**. Since F2 R2 is the inverse of R2 F2 this is another adjoint. This will bring all pairs into parity.
- Finally consider the case where all four OOPPs are in two faces, two in each of two adjacent faces. Orient the cube so that these two faces are in the up and front faces. Execute **R2 F2 CycleFivePairs F2 R2**. This is yet another adjoint that results in all pairs being in parity.

At this point all the diagonal pairs will be in parity but some may be in the wrong face. The two pairs in the front face may both be in the wrong face forming a square. The symmetric pairs in the back face also form a square. Execute

ExchangeRectangles = (R2 L2 T2)2

Another possibility is that there may be a pair in one face that should be in the opposite face. Roll the cube so that that pair is in the front face of R. There will be a pair in the back face that should be in the front face. If that pair is also in R execute

ExchangePairs = (R2 T2)2

If that pair is in L execute

B2 ExchangePairs B2

an adjoint.

The cases where there are two and six OOPPs deserve revisiting. Assume we have two OOPPs in the up and front faces of R. We have observed that executing CycleFivePairs produces a situation with four OOPPs. In traditional mathematical lingo, we have reduced the problem to a previously solved case and consider the matter settled.

But assume we have two OOPPs located in the up and front faces of R. When we execute CycleFivePairs the result is no merely four OOPPs. The result is the case CycleFivePairs was designed to handle. That is we can execute CycleFivePairs twice without a pause in between.

This observation invites a further examination of the case with six OOPPs. If we have the two in parity pairs in the up and front faces of R and execute CycleFivePairs we will get all eight pairs out of parity. Not the desired result. But when CycleFivePairs is executed once we get a configuration in which the four OOPPs are in the up face of R and the other three in the front, back, and down faces of L. Thus if we roll the cube to exchange the freewheeling faces we get the desired configuration. So in the six OOPPs case execute CycleFivePairs twice separated by the roll.

Any cube with at least six layers has face blocks, blocks that are in no edge, no diagonal, nor any center layer. Our first order of business is to create pairs of face blocks.

For this discussion assume we are working with a six layer cube. I make this assumption to simplify the discussion by assuring that all visible face blocks are involved. There are twenty such face blocks, four each red blue and green, and eight orange. We want to reconfigure them into ten pairs, two each red blue and green, and four orange.

A *cross pair* is a pair of blocks of the same color, one in L and one in R in the same horizontal layer. By rotating L and R independently it is easy to create cross pairs. A cross pair can be converted to a pair by executing

CreateAPair = T2 L2 T2

While CreateAPair will convert a cross pair into a pair it will also convert a pair into a cross pair. So when you use CreateAPair to convert a cross pair in the front face to a pair you do not want a pair lurking in the back face that gets converted into a cross pair. That is not progress.

Rotate L and R to place the cross pair in the up face. Rotate the up face 90 degrees in either direction. The only purpose served by this rotation is to get the cross pair out of L and R. Then L and R can be rotated freely without breaking the cross pair. Rotate R and L to get another cross pair. Rotate that cross pair to the down face. Restore the correct up face orientation. Rotate the R and L faces to place these cross pairs in the front and back faces and execute CreateAPair.

The operations described above can be performed in all cases except when there are seven pairs. A configuration with nine pairs is impossible, so we cannot create two new pairs when there are seven pairs. If there are seven pairs create a cross pair in the up face and rotate the up face 90 degrees in either direction, Rotate two pairs of the same color into the down faces of L and R. Restore the up face to its required orientation. Do Lu and Ru and execute CreateAPair. This gives us eight pairs. We can then create the last two pairs. We now have ten pairs, two each red, blue, and green, and four orange.

From this point on phase three proceeds the same way that phase two went.

Phase Three Repeated

If there are more than seven layers Phase Three will be repeated until there are no more active pairs. If the number of layers is even the run terminates with a solved cube when we run out of active layers.

Final Phase

For a cube with an odd number of layers and all active pairs of layers solved we are left with a center layer C with a right-left axis and the left to right center row in the up face. The blocks in play fall into two types, center edge blocks and center face blocks. They are independent and can be solved in either order.

We will take on the center edge blocks first. There are six center edge blocks and all have a red face or an orange face. A center edge block is *flipped* if its red or orange face is in a blue or green face of the cube. The following procedure will flip, or unflip, two edge blocks.

FlipTwoEdges = Cd Ul Cd U2 Cu Ul Cu

The two edge blocks that are flipped by this procedure are the down front edge of C and the up left edge of the up face. We have to be sure that these two positions are occupied by blocks that require unflipping, but we do not need an adjoint.

Despite its complexity FlipTwoEdges moves few blocks, the two flipped blocks and one other in the right edge of the up face. We must keep the orange face up but we can exchange the freewheeling faces.

If there is a flipped block in the down face and a flipped block in the up face, roll the cube to place the flipped block in the down face in the down front edge. Rotate the up face to put it in the left edge and execute FlipTwoEdges.

If there is a flipped block in the down face but no flipped block in the up face both flipped blocks are in the down face. Execute

DropEdgeBlock = Cu U2 Cd U2

DropEdgeBlocks drops the up front center edge block into the down front edge. This will give us flipped blocks in both the up and down faces.

If both flipped blocks are in the up face, rotate the up face to place a flipped block in the up front edge and execute DropEdgeBlock. This will give us flipped blocks in both the up and down faces.

Repeat until there are no flipped blocks.

We now have to place the up left and up right edge blocks. They cannot be placed one at a time. If both are not in the down layer there is one in the up layer. Locate it and rotate the up layer to place that block in the front up edge. Execute DropEdgeBlock to drop this block into the down layer. If necessary repeat until both edge blocks are in the down face. We will see one of these blocks in the down front edge. Its face will be white or yellow. Rotate the up layer so that the up front corner blocks have matching colors. Roll the cube so that the down face becomes the front face and execute **(C2 U2)2**.Restore the default orientation.

Finally we deal with the center face blocks in the center layers. The white and yellow center face blocks were placed in Phase One. The red and green center face blocks were placed at the end of Phase One. This is where placing the red and green blocks early pays off. That leaves us to deal with only the orange and blue block center face blocks.

I will use an example to illustrate how to deal with it. Locate an orange center face block in the front (blue) face. There must be a blue block in the up face that is the same distance from its center. These two blocks must be swapped. Assume the orange block in the front face is below the center block. Rotate the front face to place this block to the left of the center block. That is, execute **Fd**. Assume the blue block in the up face is in C in front of its center. Rotate the up pace to place it to the right of it center. That is, execute **Ur**. Then rotate its layer to place it in the front face. That is execute **Rd**. The orange and blue blocks are now in the sane horizontal layer. Rotate the front face to place these two blocks in C. That is, execute **Fu**. Now execute

SwapCennterFaceBlocks = (((C2 T2)2) B2)2

This will also swap the face blocks in the back face, but they are both green. This is an adjoint with **A = Fd Ur Rd Fu**.

Repeat until there are no more orange blocks in the front face. And we are finished. The cube is solved.

Summary

1. **SwapTwoCorners = Ru Ur Rd Ur Fuer Ul Fd**
2. **TwistThreeCornersCW = Ru Ul Rd Ul Ru U2 Rd**
3. **TradeVisitors = U2 Ru U2**
4. **CycleThreeEdges =**

   F2

   (U2 Ru U2 Rd)

   (roll exchanging front and up faces)

   (U2 Ru U2 Rd)

   U2
5. **CycleFivePairs = (U2 Ru)6**
6. **ExchangeRectangles = (R2 L2 T2)2**
7. **ExchangePairs = (R2 T2)2**
8. **CreateAPair = T2 L2 T2**
9. **FlipTwoEdges = Cd Ul Cd U2 Cu Ul Cu**
10. **DropEdgeBlock = Cu U2 Cd U2**
11. **SwapCennterFaceBlocks = ((C2 T2)2 B2)2**

Evar D Nering

12/12/2022

evar@nering.name