Fewest Moves Tutorial

Solving a Rubik's Cube with as few moves as possible

v2.0

Sebastiano Tronto

September 15, 2017

Preface to the Second Edition

For a couple of years I have been thinking about making a new version of this tutorial. There were small mistakes to correct, things to add somewhere and a couple of things to change, since with time I have changed my point of view on a couple of things.

I also thought of rewriting in LaTeX, because LaTeX documents look way better than anything else. The only downside of this choice is that it may be more difficult to translate: I was very happy to see that my tutorial was so much appreciated that people wanted to translate it in many different languages to make it more accessible worldwide! I would like to thank them all one by one, but I forgot most of their names.

But I have always postponed this second version. The main reason was that I din't have much time, but I also lacked motivation. At World Championship 2017 in Paris I have met many people who thanked me for this tutorial, and this gave me the motivation I needed. I remembered how nice it is to be part of such a nice community where everybody helps each other without asking anything back, and I wanted to do my part - again.

The changes since the first version are mostly aesthetical. You may have notice that this book is way longer than the first one: the difference is mostly pictures and blank pages. I have rearranged a few sentences, corrected a few mistakes (my English got better in the last 3 years!) and probably also added a few.

I have also added a couple of things:

- Section 2.3.3 *Other Edge 3-cycles*. But there is still much to be said about those cases.
- Section 2.4.8 about "3 edges and some corners" kind of insertions.
- Section 3.6 about solving with "skew centers".
- Appendices for Notation and Last Layer algorithms.

Another thing I did was to add nice boxes for example solves - see below. This is mainly because I wanted the book to be more self-contained, since many of the solves were just linked in the first version. I have kept hyperlinks, but I have also almost always written the complete link as a footnote. This makes this book suitable both to be read on an electronic device and on paper.

When I occasionally talk about the colours of the pieces we are considering, I assume you are using the standard color scheme[1] and that you scramble in the standard orientation: white on top and green on front.

I don't have anything else to say about this second edition. Go on and enjoy the tutorial!

[1] `http://www.speedsolving.com/wiki/index.php/Western_Color_Scheme`

About this Book

This book is intended to be a guide to get good results in the so called “Fewest Moves Challenge”, one of the official events in WCA competitions. If you don’t know about the World Cube Association or speedcubing competitions, see the introduction a few pages ahead.

The “Fewest Moves Challenge” is about solving a specific (scrambled) position of the Rubik’s Cube in as few moves as possible, using only some cubes, pen and paper. The use of computer programs is not allowed, and there are usually time constraints (one hour in WCA competitions, up to one week in some online ones).

In view of this, no general solving algorithm is described. The reason why it is not convenient is repeatedly remarked during the rest of the book and can be summed up as follows: restricting to a single approach is too limiting.

Likewise, there is no mention on how to program a computer to generate (short) solutions. If one is interested in the subject I suggest taking a look a the *Computer Puzzling* page at `www.jaapsch.net`.

Prerequisites

The reader is assumed to know how to solve a Rubik’s Cube, even with a beginner’s (LBL) method. Additionally, familiarity with the standard OBTM notation (link in the introduction, see also Appedix B) and with some basic terminology (for example, what is 2x2x2 block in a Rubik’s Cube) is required, although many technical terms will be explained in the text.

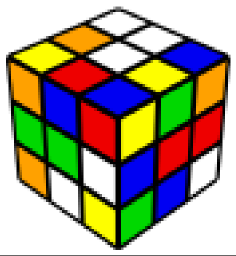
About Example Solves

In the second version of this book I have added some example solves, in order to make the text more self-contained. They appear in nice boxes like the following:

2C2E Insertion - Example

Scramble: `B' L' D2 R U F' U' L U2 D R2 U2 F B R2 B U2 B L2 B2 U2`

```
B' F D2    //Pseudo 2x2x1 (3/3)
L' B * R2    //Pseudo 3x2x2 (3/6)
F2 D F' D2 F    //Found using NISS (5/11)
R' D' R D2 F U2    //Found using NISS (6/17)
* = B2 L B L' B D2 F' R F D2    //2c2e insertion (9/26)
```



Solution: `B' F D2 L' R2 B R B' R2 F R' B R F' R2 F2 D F' D2 F R' D' R D2 F U2` (26)

See on alg.cubing.net

You may notice that in writing the solution I almost never use “rotation” moves such as `x` or `[f2]`. This doesn’t mean that you shouldn’t turn the cube around in your hands when trying the solution. If you are not familiar with rotationless solution writing, I advise skipping to Section 6.1 before reading the rest of the book.

Acknowledgements

I want to thank the whole international speedcubing community for always openly spread techniques and methods, enabling anyone to freely learn anything that has been discovered (until now). I hope this tutorial will be helpful in the same way.

I also want to thank everybody who gave me suggestions for improvements, pointed out mistakes or translated this tutorial (or rather, the old version). I don't want to name any of you explicitly, because I know I will forget many.

For this second edition I made use of visualcube[2] and alg.cubing.net[3], so a special thanks goes also to the creators of this two tools, respectively Conrad Rider and Lucas Garron.

Disclaimer

As you may know, English is not my first language. If you think that a certain passage is poorly written or if you find any mistake, please contact me at: sebastiano.tronto [at] gmail [dot] com.

[2] `http://cube.crider.co.uk/visualcube.php`, although I have acutally used the version hosted at `http://stachu.cubing.net/v/`, because of problems with png images.

[3] `https://alg.cubing.net/`

Contents

Introduction

Trying to solve a Rubik's Cube (or, in general, any puzzle) as fast as possible is interesting, but it is even more interesting trying to solve it using the fewest moves: this is the goal in "Fewest Moves" solving (or FMC, "Fewest Moves Challenge").

Official Competitions

FMC is an official event recognized by the WCA (World Cube Association), the organization that governs competitions for the Rubik's cube and similar puzzles. Official regulations (Article E[4]) can be summed up as follows:

- The scramble (sequence of moves to scramble the cube) is given to all competitors, written on a sheet of paper.
- Time limit: 60 minutes.
- Allowed material:
  - Paper and pens (provided by the judge);
  - Up to 3 Rubik's cubes (self-supplied);
  - Unlimited colored stickers (self-supplied).
- The solution:
  - Has to be submitted written in the OBTM[5] notation: allowed moves are rotations (`x`, `y2`, `[u]`...), single-layer moves (`R`, `U2`, `L'`...) and wide moves (`Rw2`, `Fw'`...) but not inner-layer moves (`M`, `E`, `S`...);
  - for the final results, rotations are not counted, while each other move counts as 1;
  - Has to be at most 80 moves long (including rotations);
  - Must not be related to the scramble in any way; in addition, the competitor must be able to explain each move in his solution.

In order to enforce the last rule, since 2016 scrambles are generated so that they begin and end with the sequence `R' U F`.

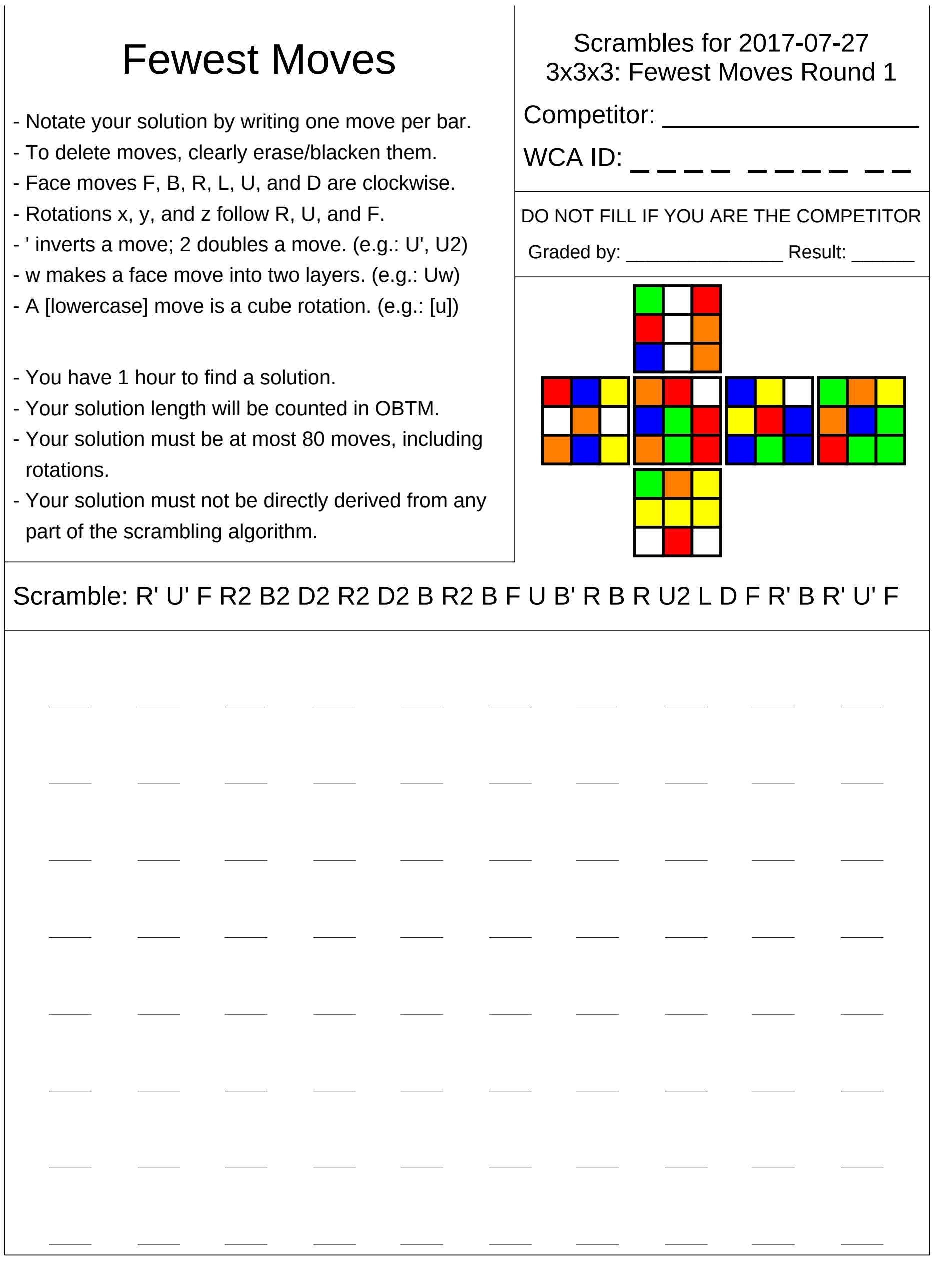
The best result ever achieved in competition is 19 moves, by Tim Wong (USA), Marcel Peters (Germany) and Vladislav Ushakov (Belarus). The world record for an average of three attempts (in one competition) is 24.33 by Marcel Peters and Baiqiang Dong (China). Marcel Peters also holds the World Champion title (2017).

Goal of This Tutorial

The goal of this tutorial is to summarize the best known techniques that lead to good results in fewest moves solving. In some cases the explanation will detailed and enriched with examples, while in some other I will only provide a synthetic explanation and suggest other resources for further study.

[4]https://www.worldcubeassociation.org/regulations/#article-E-fewest-moves

[5]https://www.worldcubeassociation.org/regulations/#12a



Fewest Moves

- Notate your solution by writing one move per bar.
- To delete moves, clearly erase/blacken them.
- Face moves F, B, R, L, U, and D are clockwise.
- Rotations x, y, and z follow R, U, and F.
- ' inverts a move; 2 doubles a move. (e.g.: U', U2)
- w makes a face move into two layers. (e.g.: Uw)
- A [lowercase] move is a cube rotation. (e.g.: [u])

- You have 1 hour to find a solution.
- Your solution length will be counted in OBTM.
- Your solution must be at most 80 moves, including rotations.
- Your solution must not be directly derived from any part of the scrambling algorithm.

Scrambles for 2017-07-27
3x3x3: Fewest Moves Round 1

Competitor: ____________________

WCA ID: _ _ _ _   _ _ _ _   _ _

DO NOT FILL IF YOU ARE THE COMPETITOR

Graded by: _______________ Result: ______

Scramble: R' U' F R2 B2 D2 R2 D2 B R2 B F U B' R B R U2 L D F R' B R' U' F

Figure 1: Example of official scramble sheet.

Chapter 1

Think Outside the Box

Whatever your main method is, forcing yourself to only use that method is the worst thing you can do in FMC. **You should never restrict yourself to only one method**, but try to exploit every situation.

For example, suppose you have built a 2x2x3 block; now you have many possibilities: you can place the last "cross" edge and finish the F2L (CFOP), you can orient edges (Petrus) or try to build more blocks freely (FreeFOP, Heise) and so on. Any of this methods may lead to a good solution, so the best thing to do is to **try to use them all** (or most of them at least).

The best way to open your mind and learn how to think outside the box is, maybe a little paradoxically, to **get into many "boxes"**, that is to learn many methods. Here I will briefly describe those that are, in my opinion, the most useful methods for FMC, without talking about their development history nor their pro/cons in speedsolving. For each of these methods I will provide a link to an online tutorial, but I suggest you look for more information about them on google or speedsolving.com. The *Beginner's Guide to Choosing a Speedsolving Method*[1] on speedsolving.com forum, although mainly thought for speedsolving, is a good starting point for getting more information about the 4 most commonly used methods (CFOP, Roux, ZZ and Petrus).

1.1 Petrus

Petrus' steps are the following:

1. Build a 2x2x2 block.
2. Expand the 2x2x2 block to a 2x2x3 block.
3. Orient edges.
4. Complete the first 2 layers (F2L).
5. Solve the last layer (originally divided into 3 steps).

Having a good **blockbuilding**[2] skill is mandatory if you want to get good at FMC, and practicing Petrus is the best way to acquire it. To learn how to solve steps 1 and 2 efficiently, you need to think and try to explore different ways of building blocks every time; it is also very useful to study **reconstructions of expert cubers' solves**. For the first step it is also helpful to compare your solutions with the optimal ones, given by an optimal solver[3], since for an expert

[1]`https://www.speedsolving.com/forum/threads/beginners-guide-to-choosing-a-speedsolving-method.43471/`

[2]"Blockbuilding" or "block-building" is the technique consisting of making blocks of pieces and then joining them together. It is often seen in opposition to the way of building the F2L used in CFOP (see below), but this can also be seen as a kind of blockbuilding. A more suitable contrast is given by comparing it to other techniques, such as edge orientation (Petrus, ZZ), "Corners First" solving, using algorithms or commutators. All of these techniques are explained in this book.

[3]For example HARCS, freely available online: `https://www.speedsolving.com/forum/threads/harcs-jarcs-replacement-cube-solver.63241/`

it should be (almost) always possible to find an optimal 2x2x2 block.

Step 3 teaches you how to recognize an edge's orientation regardless its position in the cube. This is another important skill, since flipped edges are one of the worst thing you may come across during an FMC solve. For this step, as well as for step 4, ZZ may be a better teacher than Petrus.

You don't have to learn every algorithm for solving the last layer (the same is true for other methods too). In the next chapter I will explain in detail how you should proceed. For now it is enough to be aware that the "last layer" step usually is not included in Fewest Moves solutions.

Lars Petrus Website: `http://lar5.com/cube/`.

1.2 Roux

1. Build a 3x2x1 block.
2. Build another 3x2x1 block, opposite to the first one.
3. Solve the corners of the last layer, without necessarily preserving the M layer (CMLL).
4. Solve the last six edges (LSE).

Petrus is an excellent method to learn blockbuilding, but forcing yourself to only use Petrus is still wrong: learning also Roux (especially the first 2 steps) will make your skill in some sense more complete. Also for this method it will be useful to study solutions of more expert cubers.

For step 3 it still stands what was said about last layer algorithms, while step 4 is to be avoided like plague (at least, avoid solving it in the "standard" way, i.e. using only M and U moves): remember that every inner layer move, like M, is worth 2 moves in standard metric!

Waffle's Roux Tutorial: `http://wafflelikescubes.webs.com/`.

1.3 ZZ

1. EOLine (orienting all edges and placing DF and DB, leaving the cube to be possibly solved moving only the R, L and U layers).
2. F2L.
3. Last Layer.

As mentioned earlier, recognizing and solving edge orientation is a very useful skill and ZZ is surely the best way to learn it. At first sight "orienting edges" can be hard, because it is a more abstract concept than "building blocks", but don't panic, it gets way easier with practice!

Step 2 is more or less the same as Petrus' step 4, but you have to build the blocks on both R and L side at the same time[4]. This is also going to improve your blockbuilding skills.

1.4 CFOP (Fridrich) and FreeFOP

In classic CFOP the steps are the following:

1. Cross (4 edges on the same side).
2. 4 corner/edge pair insertions.
3. OLL (Orient Last Layer).
4. PLL (Permute Last Layer).

[4]For speedsolving, it may be better to solve one block at the time, since it is usually more ergonomic. But this is not the case for FMC, as efficiency (i.e. number of moves) is the only thing that matters!

Classic CFOP s **not** considered a good method for FMC, but in some situation it is useful to know different ways to insert a corner/edge pair.

In FreeFOP, the first two steps are replaced by a “free” blockbuilding F2L.

Anyways, I strongly advise against solving the last layer with OLL + PLL, unless you get to skip one of the two steps.

Badmephisto’s Tutorial: `http://badmephisto.com/`.

1.5 Keyhole F2L

Keyhole is not really a method to solve the cube, but a technique to solve the first two layers. It is considerate an intermediate method between “Layer by Layer” and CFOP. The steps are the following:

1. Cross.
2. Place 3 first layer corners.
3. Insert 3 middle layer edges, using the “free” corner.
4. Insert the last corner/edge pair, as in CFOP or as in LBL.

To improve efficiency you should replace the first two steps with blockbuilding and place a few middle layer edges in the meantime. A variation consists in solving the cross and 3 middle layer edges, and then placing 3 first layer corners using the “free” edge.

Despite its simplicity, this method can be very useful in FMC.

1.6 Heise

1. Build four 2x2x1 “squares” (all sharing one colour).
2. Match the squares and orient edges.
3. Solve the remaining 5 edges and 2 corners.
4. Solve the last 3 corners using a commutator.

If you decided not to follow the experts’ advice and use only one method for FMC, Heise would be a good choice. This method alone can lead to and average of fewer than 40 moves for linear[5] solves. It is an extreme method, but also extremely efficient.

First two steps are a strange but efficient way of building an F2L-1[6] and orient all edges. The “don’t restrict yourself” and “exploit different situations” concepts is perfectly applied, allowing one to solve the blocks in whatever is the best way.

The third step is complex, but it is a more efficient alternative to finish the first two layers and then solving the last layer using algorithms. Practicing it will give you the ability to build and move around blocks when you are limited by the quantity of blocks already built (and this is the reason why this step is so difficult).

For the last step you will need commutators; it allows, in an FMC solve, to use insertions (both these techniques will be explained in the next chapter).

Heise method’s page on Ryan Heise’s website: `http://www.ryanheise.com/cube/heise_method.html`. There you can find not only a detailed explanation of his method, but also other useful information (see for example the *Fundamental Techniques* page).

[5]As in “linear” FMC, that is without trying different possibilities and/or cancelling or undoing moves.

[6]With “F2L-1” I mean an F2L minus a coner/edge pair.

1.7 What and How to Learn

Obviously, getting fast with all of the methods described is not your goal. Doing some speedsolves may help seeing some things faster and is fun, but **we don't care about speed**. Since our goal is to solve the cube with the fewest moves possible, you should try to be efficient. It is also essential to be **color neutral**[7] and it can be helpful trying to work with **"Non Matching Blocks"**[8].

But the main difference between speedsolving and fewest moves solving is that in FMC you can **try different possibilities**. If in Petrus, for example, you are left with 6 "bad" edges after a 2x2x3, you can try to build a different block from scratch or to slightly modify the contruction of the block you have found to improve your situation[9].

Here is some piece of advice for some of the methods described.

1.7.1 Petrus

- After completing a 2x2x2 block, you can expand it in **3 different directions**. Make sure to consider all of them!
- Try to build a 2x2x3 directly, instead of going through the 2x2x2 step.
- Try using Non Matching Blocks in step 4.
- In step 4 again, try influencing the last layer to get an easier case (even "Heise style").

1.7.2 Roux

- Try to build the two blocks at the same time.
- Try Non Matching Blocks.
- Influence the CMLL while finishing the second block, and the LSE during the CMLL.

1.7.3 ZZ

- After edges orientation, building the "Line" isn't always a good idea: try blockbuilding directly (without breaking the EO!). The difference between solving the line first and then the right and left blocks is comparable to the difference between doing CFOP with cross + F2L pairs and doing FreeFOP.
- Once you have oriented the edges, you have reduced the cube to be solved moving only the R, L ,U and D layers: you can build the F2L on any of these sides.
- As in Petrus and Roux, try using Non Matching Blocks and influencing the last layer while building the F2L.

[7]Being able to solve the cube by starting with any "colour"; for example, starting from any cross in CFOP or from any of the 8 possible 2x2x2s in Petrus.

[8]Sometimes also called "Pseudo Blocks", especially in FMC. It is a useful technique in Roux, ZZ and Heise, but it can be used in other methods as well. It consists of building blocks different from the ones you should build if you are following the method, but that can be placed in the same "slots". For example, in Roux, the second 3x2x1 block can be any of the 4 that can be built on the side opposed to the first one. This technique is very powerful if combined with premoves, that will be explained in Chapter 3.

[9]Trying to influence a later step while solving another the current one is a good habit, which will be discussed again later.

1.7.4 CFOP/FreeFOP

- **FreeFOP is better than CFOP**, at least because CFOP is a special case of FreeFOP, by definition. Try at least to build and **XCross**[10].
- Try to influence the last layer edges' orientation, avoid the "4 edges flipped" cases; some ZBF2L algorithms can be useful, but instead of learning them by heart try to **study them to understand how they work**.
- Some optimal pair insertion algorithms are not well known (for example `F2 U R U' R' F2`): study them, again trying to understand them rather than learning them.
- Try **"multislotting"**, that is inserting more pairs at the same time. The easiest case is when you use a D-layer move as setup, for example `D R U R' D'`. There are algorithms for this technique available online, but I suggest trying in a "free" way: for example, look at how I have inserted the second, third and fourth pair in this solve: `https://www.speedsolving.com/forum/threads/the-3x3x3-example-solve-thread.14345/page-238#post-1013053`.

[10]Cross and first pair, built at the same time. It can also be seen as a 2x2x2 block + 2 edges.

Chapter 2

How to Proceed During a Solve

To quote Per Kristen Fredlund, the general way to proceed is the following:

"*Think of it more like so: solve the cube in 2 stages where stage 1 solves as many pieces as possible as efficiently as possible (i.e.: make a good skeleton*[1]*). The second step is fixing the unsolved pieces, typically by inserting algorithms into the skeleton*[2]*".*[3]

This is the general approach, but you don't need to use it every time: sometimes you can find a very short solution by, for example, solving the F2L with blockbuilding and then finishing the last layer with an algorithm. There are also different ways to proceed, two of which will be explained in Chapter 4.

If this description seems too generic, it is because it can't be differently: there isn't a standard method that allows you to always get good results, **you should always try as many strategies as you can**, so that you don't miss any chance.

Here I will describe some basic techniques used in FMC solves. You have already learnt some of them by practicing the methods described in the previous chapter, while you will need to study some other. Some will be explained in detail, some other will be only mentioned and other tutorials will be linked for a more complete study.

2.1 Blockbuilding

Blockbuilding is probably the most important technique in FMC. It is a simple concept, but it requires a lot of practice to be mastered. Practicing blockbuilding-based methods (Petrus, Roux, Heise and ZZ), in the ways previously described, is the most direct way to improve at blockbuilding.

Here I will list some fundamental techniques that will come in handy; the first ones are taken from Ryan Heise's website (`www.ryanheise.com/cube`), which is full of examples: look them up!

2.1.1 Align then Join

Source: `http://www.ryanheise.com/cube/align_join.html`

Basic technique: to build a corner/edge pair, the simplest kind of block, you have to align them first, making them joinable by a single move, and then join them with that move.

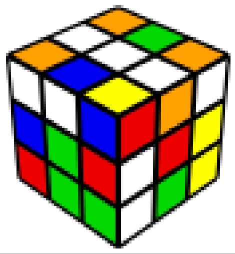
This concept can be applied, in a more general meaning, to the case of building bigger blocks, for example when you want to match a pair and 2x2x1 square to get a 3x2x1

[1]A partial solution, where only a few pieces (usually from 2 to 6) are left to be solved.

[2]Technique that allows to solve a few pieces by inserting moves somewhere earlier in the solve. It will be explained soon, be patient!

[3]`http://www.speedsolving.com/forum/threads/fewest-moves-tips-and-techniques.1566/#post-16209`

All of this may look trivial, and in some way it is; the important thing to learn is to **recognize when two pieces are aligned** and can be therefore joined with one move. This way you will be able to realize in advance whether a certain move will join 2 pieces.

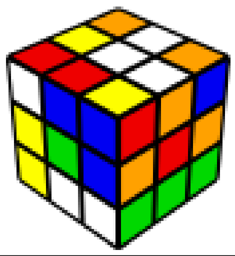
Align then Join - Example
Scramble: `F U' D2 L D L2 D F R2 B U2 R2 D L2 D L2 D R2 U' L2 F2`
`L2` //Align `U2` //Join 
See on alg.cubing.net

In the example above, two pairs are already built. The sequence `L2 U2` pairs up the blue-red edge with the blue-red-yellow corner.

2.1.2 Move it out of the way

Source: `http://www.ryanheise.com/cube/move_it_out_of_the_way.html`

It can happen that we want to build a block, but the move required to build it breaks other blocks, or moves away pieces that we don't want to move. One of the ways to bypass this problem is to move away such pieces first, saving them from the breaking move, and then, if necessary, put them back together once the move has been performed.

Move it out of the Way - Example
Scramble: `F R2 B D2 F D2 L2 B2 F' D2 F' L B U' F2 U2 L2 U B R2 F`
`U2 R2 D R2` //2x2x1 square `U'` //Move the red-white-blue pair out of the way! `F' D'` //Expand the square to a 2x2x2 block 
See on alg.cubing.net

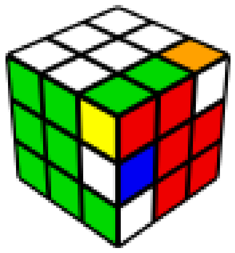
If we did `F' D'` right after building the 2x2x1 block we would still get the same 2x2x2 block, but we would break the red-white-blue pair. Notice however that in this case "moving it out of the way" doesn't seem to be the best idea: in the process of saving that pair, we are breaking the yellow-orange-blue one!

2.1.3 Destroy and Restore

Source: `http://www.ryanheise.com/cube/destroy_restore.html`

Another approach to solve this problem is to temporarily break some blocks and join them back later, using the "move it out of the way" technique.

A basic example is given by applying the "scramble" `R U R' U'`.



Let's ignore the last layer and pretend not to know that `U R U' R'` obviously solves the cube. Looking at the first two layers only, we see that `R'` places the ready pair next to the other pieces in the front layer, but breaks part of the F2L already built. We can use "Destroy and Restore" this way:

```
R'  //"Destroy"
F  //"Move it out of the Way"
R  //"Restore"
F'  //Put back the pieces moved away with F
```

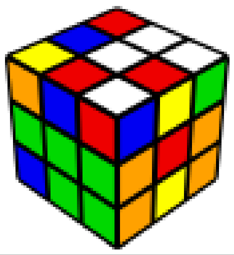
2.1.4 Keyhole

We talked about it earlier as a standalone method, but keyhole can be considered a particular strategy to build blocks. This technique consists in exploiting unsolved parts of the cube to solve some pieces. After some good Keyhole F2L practice you shouldn't need any example to understand what I mean, but I'll leave here a Keyhole solve anyways.

Example (adapted from a solve by Edoardo Disarò)

Scramble: `U' R' L F' B U2 R2 B2 L' B R D F2 D2 L2 F2 D' R2 F2`

```
F' L'  //Layer minus one corner
F2 L' B' L  //Keyhole
F U' B U  //Keyhole, accidentally solving the last corner
F' R' B2 R  //Keyhole
F B L B L'  //F2L
B'  //LL
```



See on alg.cubing.net

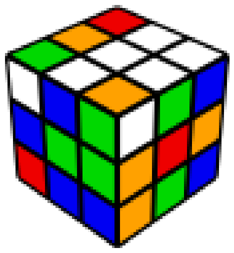
2.1.5 One Move, Two Goals

It is often possible to use only one move to build two blocks or, in general, to "get two things done". An example will make this clearer.

Example (by Mirek Goljan and Guus Razoux-Schultz[4])

Scramble: `D U' F2 U' R' F R2 B D' B R F B' U R' D2 L' R2 F2 B' U' B D B2 F2 U L F U' B2`

```
L U' F2 D'  //2x2x2 (4/4)
U2 B R2 B  //Pseudo F2L-1 (4/8)
F' * U F R U2 R'  //Pseudo F2L (6/14)
U2 R2  //All but 3 corners (2/16)
* = B' U F2 U' B U F2 U'  //Last 3 corners (3/19)
```



Solution: `L U' F2 D' U2 B R2 F' U F2 U' B U F' R U2 R' U2 R2` (19)

See on alg.cubing.net

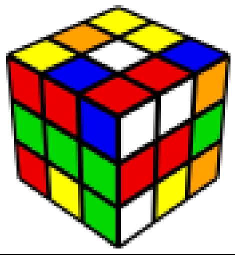
If you don't know about insertions yet, ignore the last line. In fact the only line we are concerned with is the first one, especially the red `F2` move: notice how that single move builds the 2x2x1 block in DF and places the orange-green edge at the same time, which allows to build the 2x2x2 with the next move.

Situations like this may arise without forcing them, but it useful to learn to recognize them, in case they don't.

2.1.6 Influence Later Steps

We have already (quickly) talked about influencing the LL step while finishing the F2L[5]. This idea also applies to blockbuilding: it is often better to build a block sub-optimally[6], or to add some "unnecessary" moves to have a better continuation, blockbuilding or else (i.e., edge orientation).

For example, consider the following scramble.

Influence Later Steps - Example	
Scramble: `L2 D2 U R2 F2 D2 B2 U' R2 B2 U B U F D B2 U L D' R' F`	
`L2 B R B` //Two 2x2x1 blocks	
See on alg.cubing.net	

As you can see, the sequence `L2 R B` builds the red-blue-white square, but adding only one move (the red `B`) the square become 2.

2.1.7 Pay Attention to EO

Here EO is the common shortcut for Edge Orientation.

Someone may have noticed, while studying different methods, that "Edges Orientation" is a recurring step. As said before, **the more bad edges there are, the harder things will be**. Orienting edges at the end is usually not efficient. Orienting them first, as in ZZ, is convenient, but it can be limiting for the blockbuilding phase.

One of the best things to do is trying to solve, at least partially, edges orientation during the blockbuilding step. Experience with methods such as ZZ and Petrus can lead to being able to easily spot if an edge will be correctly oriented after some moves. If you don't have this ability yet, remember that in FMC solves you can go back and modify your solution every time you wish: if you have troubles with EO, try going back and add/change some moves to see if things get better (see also Section 2.5.1).

However, don't dismiss the "EO first approach" too quickly: notable cubers such as João Pedro Batista Ribeiro Costa (2015 World Champion) and Grzegorz Łuczyna (2010 European Champion) almost always start with edge orientation and other like Sébastien Auroux (2011 World Champion) and myself do it very often. For more details and examples see section 4.1.

2.1.8 Which Block Should I Build?

The golden rule is to exploit different situations: a 2x2x2 block, a 3x2x1, two 2x2x1 squares and many other kind of blocks can be a good start. Try out every possibility.

[5]Notable examples are ZBLS (sometimes less properly called ZBF2L) and Winter Variation, but there are many more: look them up!

[6]"Sub-optima" (with or "suboptimal", without the -) refers to a solution (complete or partial) that uses more moves than the best possible one.

Two possible approaches are:

1. Try completing big blocks, like a 2x2x3 or a F2L-1.

2. Proceed in small steps, building many small blocks and joining them at the end.

Erik Jernqvist proposes the following number of moves as good starts[7]:

Type of Blocks	Number of Moves
2x2x1 square + corner/edge pair	3
2x2x2 block	4
Two 2x2x1 squares	5
2x2x3 block	9
F2L-1	14
F2L	17

Personally, I think these are good estimations, especially for the first 3 cases. But always have to remember that **how good a start is depends on the possible continuations**. If you build an F2L-1 in 12 moves, but you have 4 bad edges, one of which is the last F2L edge flipped in place, you may have to throw it away. On the other hand, a 2x2x3 in 12 moves with all edges oriented can be a better alternative. Obviously EO is not the only thing you have to consider, but it's one of the most important.

Another rule is: **never[8] complete the F2L without influencing the last layer**. The reason is simple: a bad last layer can require many moves, no matter how many algorithms you know, and a completely built F2L gives a little freedom to manipulate the cube. On the other hand, **an F2L-1 is a good partial goal**, because it leaves you more with more freedom, despite the many pieces already placed.

2.1.9 Ready-Made Blocks: How to Deal with Them?

It can happen that you find some blocks already built after scrambling the cube, or that with the first moves you unintentionally build some more blocks (mostly corner/edge pairs). In this cases it is preferable to exploit those blocks rather than giving them and up and going on your own way. How? There are three ways:

1. Expand / match the ready-made blocks first; the obvious thing to do.

2. Expand / match the ready-made blocks, but **watching other pieces too** and trying to build other blocks at the same time; this is usually the best thing to do.

3. Don't expand the ready-made blocks, but try to make new ones, possibly saving the others.

Moreover, it is very important, but terribly difficult, to **understand when it is not worth to save a block**, and you should break it to make new ones. Personally, I find it very hard to "give up" and "throw away" a block, but I realize this often causes troubles, especially in time management.

[7]Taken from this post on speedsolving.com. Obviously, what a "good start" depends on your level. The given move counts are to be considered a good goal for someone who wants to become an expert. If you are not this good yet you can go on with less efficient blocks. Don't waste too much time looking for a good start: it's the final result that counts!

[8]But a more important rule is: never say never!

2.1.10 Tricks to Get Fast and Advanced Techniques

"Getting fast" is not to be intended as in speedsolving, but as "finding faster a good blockbuilding start[9]". Being fast at finding a good start is very important, because it saves time (one hour isn't that long!) and you should try to explore, or at least to notice, every promising start.

The simplest kind of block is the corner/edge pair, or 2x1x1 block. There will probably be one already made in a scramble, without making any move. If there is one (or more than one), you can deal with it in one of the ways described in the previous section 2.1.9. If there isn't any, you should learn how to recognize at the sight pairs that can be made with one move (see 2.1.1 "Align Then Join"). If you are not fast at recognizing them yet, or if you want to avoid some effort[10], you can use the "Brute Force" strategy: try all possible moves, starting with `U`, `U2`, `U'`, then `R`, `R2`, `R'`, and so on, checking after each move if you did build a pair.

A more advanced technique, more useful but requiring more thinkin, consists in checking every possible 2x2x2 (there are 8 of them). You can do it this way: **for each corner, look for its three matching edges** and try to see which way you can join them to make that 2x2x2 block. Try not to make any "test" moves, so that you can do the same for the other corners without re-scrambling. Usually, if I see a very bad 2x2x2 block (one that requires too many moves) I just ignore it and go on. This technique, beside enabling you to find, most of times, an optimal 2x2x2 block (which is usually a good starting point), gives you an idea of where every piece is on the cube.

Another reason why I suggest trying to "see" every move for making the 2x2x2 without performing is that to get faster and better at blockbuilding it is useful to be able to "calculate"[11] pieces' movements without seeing them. Alexander Lau (2014 Rubik's Cube European Champion and Roux method master) is able, in the 15 seconds given to inspect the cube before a speedsolve, to plan a whole 3x2x1 block (Roux first step). His planning is so accurate that while solving this block he can look-ahead[12] and (partially) plan the second block.

You can train your planning ability with this game: ask a friend to scramble your cube with 3 moves and then try to find those 3 moves[13] (it should be easy). Then try with 4 moves, and so on. Depending on your level, you may find it difficult once you reach 6, 7 or 8 moves. If you get without problem to 9 or 10 moves, congratulations!

2.2 Find a Good Skeleton

Once you have reached good point (i.e.: and F2L-1) with blockbuilding, it is hard to go on with the techniques just described. Your goal now is to find a so-called "skeleton", i.e. a partial solution that only leaves a few pieces unsolved. The best situation is the one with 3 corners unsolved, where the pieces form a 3-cycle[14], but also 4 or 5 corners or 3 edges can be good, depending on how many moves you need to build the skeleton.

What should we do then, if the blockbuilding techniques we have seen are difficult to use without destroying what we have just built? Heise is an excellent starting point. And I mean both the method and the website: step 3 is accurately described, in particular the "Two Pairs Approach": see `http://www.ryanheise.com/cube/two_pairs.html`.

Heise's step 3 requires not only an F2L-1, but also all edges to be oriented. If they are not, you can:

1. orient them now; usually not recommended, unless it takes very few moves;

[9]A "start" can be a 2x2x2, a 3x2x1, some smaller blocks or, in some cases, a 2x2x3; in general, anything from the first 2 to the first 7 moves.

[10]Since you need to keep thinking for an hour, avoiding efforts is a good habit.

[11]The word "calculate" should be intended as in chess: a player is said to "calculate" 6, 7 or 8 moves if he can think of the possible moves and counter-moves for a total of 6, 7 or 8 moves.

[12]In speedsolving, "look-ahead" (with or without the -) is the ability to think about later step while you are solving another one.

[13]You can choose between HTM, QTM or STM, but agree on which metric to use first!

[14]A 3-cycle is a cycle of 3 pieces. For example, the A and U perm PLL cases are 3-cycles, as well as the algorithm `L F R F' L' F R' F'` (Niklas).

2. modfy the steps you have already solved, to get all edges oriented at the end;

3. orient them while finishing your skeleton.

To sum it up, the possible approaches to get a skeleton are many and the best way to train in this case (and always a good habit) is to look online (for example in websites hosting online competitions or on speedolving forums) for expert FMCers' solves; most of times, they build a skeleton.

Some deliberate practice can also be helpful: on qqTimer (`http://www.qqtimer.net/`) you can choose the scramble type 3x3x3 subsets → last slot + last layer.

You obviously don't have to build an F2L-1 before completing your skeleton, but is often the easiest way. In any case, try to save small blocks (pairs or 2x2x1 squares) made by LL pieces, if some of them happen to get built.

2.3 Commutators

According to speedsolving's definition[15], a commutator is a sequence of moves of the form

`A B A' B'`

where `A` and `B` are move sequences and `X'` is the inverse sequence of `X`[16]. Such a commutator can be written in compact notation as

`[A, B]`

The name and notation for commutators come from Mathematics, in particular from Group Theory. See `https://en.wikipedia.org/wiki/Commutator`.

For example taking `A`=`R` and `B`=`U` realizes the "sexy move" as a commutator:

`[R, U]` = `R U R' U'`

Taking instead `A` = `R` and `B` = `U' L' U` gives the "Niklas":

`[R, U' L' U]` = `R U' L' U R' U' L U`

Often, in practice, "commutator" is used as "commutator that solves a 3-cycle". So for example the "sexy move" is not usually regarded as a commutator, while the "Niklas" is. In what follows, we will use this meaning too.

In contrast with blockbuilding, that solves a lot pieces but heavily influencing the rest of the cube, commutators solve fewer pieces (usually 3) leaving all the others where they were. Therefore, together with blockbuilding and insertions (which we will see in the next section), commutators are the basis for a good FMC solve.

2.3.1 Corner Commutators

Corner commutators are the most useful kind of commutators in FMC. We have already seen the "Niklas" as a commutator. Also the A perm PLL case `R2 B2 R F R' B2 R F' R` is (almost) a commutator:

`R2 B2 R F R' B2 R F' R` = `R2 [B2, R F R'] R2`

[15]`http://www.speedsolving.com/wiki/index.php/Commutator`

[16]For example, the inverse of `U R` is `R' U'`, not `U' R'` or `R U`!

where we have cancelled[17] the sequence `R' R2` to `R`.

The 3 corners to be permuted need not be on the same layer: `[L D' L', U2]` = `L D' L'U2 L D L' U2` is also corner 3-cycle!

To learn all kinds of corner commutator (which I will *not* explain in this tutorial) is advise looking up Brian Yu's tutorial on speedsolving.com. It consists of both a written part and some videos, and it is very well made.

For FMC you only need to know "pure" 8 moves commutators. For example, the Niklas is a pure commutator, while the A perm is not. Take a look at A9s and other cases if you want, but, as we will see when talking about insertions, you will almost never[18] need them in FMC.

2.3.2 Edge Commutators

Once you have learned corner commutators, it should not be hard to understand how edge commutators work, especially the ones like

`[U R U', M']` = `U R U' M' U R' U' M`

Sadly, commutators like this only occasionally give a good result, because they use M layer moves, which are actually two moves in our metric.

The edge commutator *everyone should know* is:

`[M', U2]` = `M' U2 M U2`

Which is also very useful with some setup move, for example:

`[U: [M', U2]]`[19] = `U M' U2 M U2 U'` = `U M' U2 M U`

Remember that in official competitions you can't write a inner layer moves, so the `[M', U2]` commutator becomes:

`M' U2 M U2` = R' L x U2 R L' x' U2 = R' L F2 R L' U2

Notice how the first two moves (`R' L`) can be swapped. This is true for every pair of parallel (i.e. opposite) moves.

Another thing you need to notice is that the first two moves don't affect any of the 3 edges we want to cycle: `R' L F2 R L' U2` is therefore equivalent to `L F2 R L' U2 R'` , to `F2 R L' U2 R' L` and, since we can invert `R'` and `L`, to `R' F2 R L' U2 L`.

These observations are particularly useful when you want to cancel moves, that is making the first moves of our commutator (or, in general, any sequence of moves we want to apply) correspond to the inverse of the preceding moves (or that the last moves correspond to the inverse of the following ones).

2.3.3 Other Edge 3-cycles

There are also edge 3-cycles that don't use inner layer moves. In fact, there are some that are not even commutators! Here are two that are 8 HTM long:

`R2 Fw2 R2 U R2 Fw2 R2 U` = `R2 B2 L2 D L2 B2 R2 U`
`R2 Fw2 R2 Uw R2 Fw2 R2 Uw` = `R2 B2 L2 U B2 R2 F2 D`

[17]In a sequence of moves, we say that one or more moves *cancel* if there are consecutive moves that can be merged together (as in our example `R' R2`=`R`) or that are one inverse to the other (such as `L L'`). For example, if our F2L ends with ...`U R2 F'`, and we want to use the algorithm `F R U R' U' F'`, 3 moves cancel: ...`U R2 F F' R U R' U' F'` = `U R' U R' U' F'`

[18]Even in this case there are exceptions: see for example this post by Tomoaki Okayama and the following discussion: `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-21#post-440873`.

[19]The notation `[A:B]` = `A B A'` stands for a *conjugate*. The sequence `A` is usually called "setup moves".

Notice how the first two moves of the first one don't affect our 3 edges: we can therefore "shift" it, as we have done before with `R' L F2 R L' U2`.

But the jungle of edge 3-cycles is more intricated than this. For example, check out this 10 HTM algorithm:

```
U L D R F R' D' L' U' F'
```

and some 10 HTM 2-gen 3-cycles[20]

```
U R U R U R' U' R' U' R'
U R U R U' R' U' R' U' R
U R U R U2 R' U' R' U' R2
```

2.3.4 Block Commutators

If you have read Ryan Heise's website carefully, you already know something about the so-called "pair 3-cycles", or block commutators. If you know corner commutators, it will not be hard to find them intuitively. For example:

`[L Dw' L', U']` = `L Dw' L' U' L Dw L' U`

They are very useful in Heise's third step, but also to solve a corner 3-cycle and an edge 3-cycle at the same time. For example, the last layer algorithm `M F U F' U' F' L F R'` can be seen as:

`[R: [L' Dw L, U']]` = `R L' Dw L U' L' Dw' L U R'`

That is a pair commutator with a setup move.

The J-perm PLL can also be seen as a pair 3-cycle:

`[R2: [Fw2, D B2 D']]` = `R2 Fw2 D B2 D' Fw2 D B2 D' R2`

2.4 Insertions

After mentioning them multiple times throughout this tutorial, it is (finally) time for looking at insertions in detail.

We have somehow found a good skeleton; let's suppose we need to solve a corners 3-cycle to finish. What do we do now? We can obviously directly solve the cycle with a commutator. But, if you have studied all the cases, you know that a corner commutator can need up to 12 moves. Knowing that the best case only needs 8, those 4 more moves are not really nice. But there is a way to solve a corners 3-cycle almost certainly with fewer than 8 moves: insertions.

2.4.1 Simple Insertions

The idea behind insertions is not too difficult: if there are only 3 corners left to solve, I can go through my whole skeleton move by move and solve them when the corresponding commutator only requires 8 moves. Since that commutator doesn't affect anything but the pieces that need to be affected, the rest of the solution will solve every other piece, as it did before, and those 3 corners will also be solved, since I have cycled them with the inserted commutator.

This is how to solve almost always a 3-cycle of corners with 8 moves. How can we improve this? Among all possible inserted commutators that solve our 3-cycle, we simply choose the one that **cancels the most moves**. Uusually, it is enough to just check the cases where our 3 corners can be solved with a "pure" commutator, and then choose the best one. It happens extremely

[20]For these cases, this thread by JackW on speedsolving.com gives some explanation: `https://www.speedsolving.com/forum/threads/2-gen-edge-cycles.56224/`

rarely that the best insertion is given by a 9 move commutator (or longer), but situations like this are so unlikely that it is not worth to check every possible type of commutator.

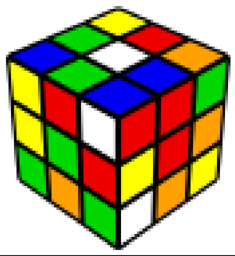
In order to make it easier to follow the movements of the 3 corners while going through your skeleton, many suggest stickering the cube with white stickers[21] on which writing numbers 1, 2 and 3 (or letters A, B and C if you prefer)[22]. Personally, I suggest taking a cheap cube with non-bright stickers and writing on it with a pencil.

An example solve will make everything clearer. Try the following skeleton on your cheap cube (or any cube, if you use stickers).

Simple Corner Insertion - Example (Skeleton)

Scramble: `D B2 U' F2 L2 D2 R2 U F2 U2 L2 R' D2 B L' U' R2 F2 R B F2`

```
B' U' D L' F'   //EO + blocks
D2 L2 D' L   //Pseudo 2x2x3
U2 R2 U' R'   //Pseudo 2x2x1
U L' U R' U' L U2 R' L'   //All but 3 corners
```



See on alg.cubing.net

At this point, grab a pencil and write "1" on the red sticker of the blue-red-yellow corner, "2" on the orange sticker of the blue-yellow-orange corner and "3" on the white sticker of the orange-blue-white corner.[23] Solve the cube (for example with `L B2 L F L' B2 L F' L2`) and re-scramble it.

We could solve the three corners right at the beginning, but we would need 9 moves (`R2 F R B2 R' F' R B2 R`). So lets perform the first move of the skeleton (`B'`) and see if we have a better case: no, still 9 moves. Perform the following move (`U'`)[24] we can now solve our three corners with an 8 moves commutator (`L2 F R F' L2 F R' F'`)! If we wanted to use it, the final solution to submit would be:

```
B' U' L2 F R F' L2 F R' F' D L' F' D2 L2 D' L U2 R2 U' R' U L' U R' U' L U2 R' L'
```

Try it out and convince yourself it works.

Anyways, we can do better. Perform the next move (`D`) and notice that this case requires 9 moves[25]. Go on this way for some more move, until you reach ...`L' F' D2`. We can again solve our three corners with 8 moves (`D' F2 D B2 D' F2 D B2`)[26]. But there is more: the last moved performed and the first of our commutator cancel! If we wanted to solve the three corners now, we should write:

```
B' U' D L' F' D2 D' F2 D B2 D' F2 D B2 L2 D' L U2 R2 U' R' U L' U R' U' L U2 R' L'
```

Which is equivalent to:

```
B' U' D L' F' D F2 D B2 D' F2 D B2 L2 D' L U2 R2 U' R' U L' U R' U' L U2 R' L'
```

[21]This is why you are allowed to bring with you "unlimited colored stickers" at a competition.

[22]Sticker need to be attached so that the 3-cycle that moves 1 to 2, 2 to 3 and 3 to 1 is the one that solves the cube.

[23]There are many equivalent enumerations: you can start from the corner and from the sticker you wish, you just have to be *coherent*.

[24]Watch out: when in a skeleton there are two consecutive parallel layers moves try swapping them too see if this gives better insertions. This is not the case, but you never know.

[25]The last 2 moves (`U' D`) are equivalent to an inner-layer move (`E'`), so they don't affect corners: it is reasonable that, before and after these moves, our 3-cycle is the same, modulo a rotation (`y'`/`y` in this case).

[26]Also `B2 U' F' U B2 U' F U`

One move less, yay! So we got to solve 3 corners with 7 moves.

For the sake of completeness, we should continue until the end of the skeleton looking for better insertions. Actually, the best insertion is towards the end:

```
B' U' D L' F' D2 L2 D' L U2 R2 U' R' U L' * U R' U' L U2 R' L'
                   * = L F' L' B L F L' B'
```

The notation above means that you should replace the `*` in the first line with sequence of moves in the second line. The final solution is

B' U' D L' F' D2 L2 D' L U2 R2 U' R' U *L' L* F' L' B L F L' B' U R' U' L U2 R' L'

Where `L` and `L'` obviously cancel, so:

```
B' U' D L' F' D2 L2 D' L U2 R2 U' R' U F' L' B L F L' B' U R' U' L U2 R' L'
```

Without further explanation you should understand how to find insertions for edges 3-cycles. You do the same also for double edges 2-cycles, solvable with one of the following algorithms:

```
M2 U2 M2 U2
R2 U2 R2 U2 R2 U2
U2 L2 D2 R2 D2 L2
```

and variations (try shifting)[27].

One last tip: 180 degrees moves (such as `U2`) can be found at the beginning or at the end of an 8-moves corner commutator only when they are the *interchange move*, i.e. when they swap two cycle pieces on the same layer. This fact can be used to seva some time: if you are aiming at cancelling two or more moves (and you should be) you can assume such a move won't cancel completely, unless it swaps two of you pieces, and only look for commutators that cancel with moves after (or before) that one.

2.4.2 Multiple Insertions: Separated Cycles (3 Edges and 3 Corners)

A skeleton doesn't have to always leave one 3-cycle: insertions can also be used to solve more (or longer) cycles.

As we have already seen, two 3-cycles, a corner 3-cycle and an edge 3-cycle, can be solved with a **pair commutator** (with setup moves, if necessary). Another way is to solve the edges with a "Sune" (`R U R' U R U2 R'`) or a variation of it, so that the only corners affected are the ones we need to cycle[28]. Both these ways should be kept in mind, but they are rarely easy to use. The "standard" solution is to insert **two commutators**.

After having numbered both the corners and the edges[29], proceed move by move as you would do with simple insertions, but at every spot you look both for a solution for corners and one for edges, besides checking for pair commutators and Sunes. Once you are done, you can write down the final solution with two insertions, but you can also **do another pass**: if you want to keep, for example, the corner commutator, but you want to look for a better edge cycle, you can write down your solution with only the corner commutator inserted. What you have now is a **new skeleton**, a few moves longer, which only leaves 3 edges. At this point you can solve them with one simple insertion. Why should we do this, when we could get anything to work with only one pass? Because a good place to insert the edge commutator can be **inside the other commutator**. You can also insert the edge cycle first and then look for a corner insertion.

The following solve is a simple example.

[27] You can also solve this case with a double insertion, as with corners double 2-cycles, see later sections.

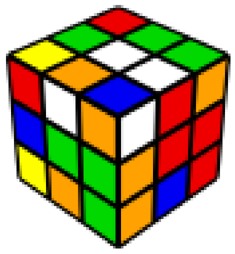
[28] Which still need to be solved somehow, possibly with another insertion.

[29] I prefer to use numbers for corners and letters for edges, so that I can't mistake one for another.

Separated Cycles Insertions - Example

Scramble: `B2 D' R2 D' F2 R2 D B2 U' L2 D2 R' U' R L' D' F D2 B R U2 R'`

`B' R' * F2 U F2` //2x2x1 (5/5)
`R` //Another 2x2x1 (1/6)
`F R U2 F R B R + B' R` //F2L-1 + pair (9/15)
`D' B' L B L'` //All but 3 edges and 3 corners (5/20)
`* = U B U' F2 U B' U' F2` //3 corners, 4 moves cancel (4/24)
`+ = R B' F D' B' D B F' R' B` //3 edges, 5 moves cancel (5/29)



Solution: `B' R' U B U' F2 U B' F2 R F R U2 F R B R2 B' F D' B' D B F' D' B' L B L'` (29)

See on alg.cubing.net

You can use exactly this approach for other cases that require you to solve two or more cycles (3 cycles or double 2-cycles) that are separated. In all other cases, things get a bit more complicated.

2.4.3 Multiple Insertions: 2 or 3 Twisted Corners

When you need to twist 2 corners, you can try to insert the commutator `[F L' D2 L F', U2]` (12 HTM) somewhere; for three corners, you can use `U' B U' F U2 B2 D' R2 U D F' U' B` (13 HTM). But this is usually not the best way, especially in the second case.

The classic way to solve 2 or 3 twisted corners is to use **two inserted corner commutators**. The first one only needs to cycle in any way the three twisted corners (or the two twisted corners + one random corner) and will usually lead to many cancellations. Then you only need to insert a simple corner commutator in the new skeleton you have got.

To do this, you only need to draw an X or any other symbol on the twisted corners; if you want to try to use one of the "pure flip" algorithms written above, I suggest drawing an arrow, so that you can tell which way you need to twist your corner (clockwise or counterclockwise).

After finding the first cycle, erase the symbols you have drawn and number the stickers 1, 2 and 3.

You can do the same with two flipped edges, but I suggest avoiding such a case, because edge commutators usually require more moves.

2.4.4 Multiple Insertions: 4 Corners

If you have 4 corners left, the only bad case, requiring three inserted commutators, is when the corner are all placed but twisted. Try to avoid it, but if you can't (or if the skeleton is really short), you can proceed as in the previous section for te first insertion, in order to reduce to a better 4 Corners case. About this situation, there is a nice discussion on speedsolving.com[30].

There are three other cases:

1. One corner is placed but twisted, the other 3 form a "twisted 3-cycle", i.e. a 3-cycle if you only regard permutation, without considering orientation. A twisted cycle always depends on some other twisted piece (or cycle).

2. Two pairs of corners that need to be swapped, correctly oriented (double swap or double 2-cycle). This case can be solved, besides using the method that will soon be described, also by transforming the corner double swap in an edge double swap: remember that an H perm (M2 U M2 U2 M2 U M2) can be transformed in a corner permutation with only one move (U2). Another way it so use algorithms such as `(R U R' U')*3` ("triple sexy") or `(R' F R F')*3` ("triple sledge"), that swap two corner pairs.

[30] `http://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-126#post-1009830`

3. A “twisted double swap”.

All these cases can be solved with **two commutators**: the first one has to solve one of the 4 corners, freely affecting the other 3, while the second one, which should be inserted in the new skeleton obtained after the first insertion, has to solve the 3 corners left. For the first step, you have only one morelimitation in the first of the three cases: the “twisted in place” corner *has to be affected* by the first commutator. If it isn’t, you are going to end up with two or three twisted corners, still requiring two insertions.

To do so, I mark the corners as follows, depending on the case:

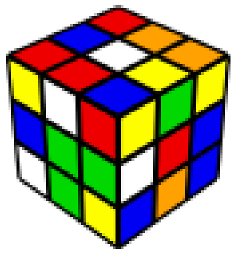
1. 1. I mark the twisted corner with and X (I don’t care which way it needs to be twisted), then I number the other three from 1 to 3. At this point, since the cycle is “twisted”, 1 belongs to 2, 2 to 3 while 3 doesn’t belong to 1, but to another sticker on that corner[31]. No problem: just mark that sticker with a 4.

2. I mark the first pair of corner with two Xs and the second one with two As.

3. Since we needed 4 numbers for a twisted 3-cycle, for a twisted 2-cycle we will need 3: reasoning as in the first case, number one of the 2-cycles 1 to 3 and the other A to C.

In the first case, for example, a possible “first cycle” is the one that sends X to 3, 3 to 4 and 4 to X. The cycle $1 \to 2 \to 3 \to 1$ is not good, because it leaves 2 twisted corners instead of a 3-cycle.

An excellent example is the following solve, previous South American Record by João Pedro Batista Ribeiro Costa. I have slightly modified the explanation of the solution: the original one made use of a *premove*, a technique treated in the next Chapter.

Former SAR (by João Pedro Batista Ribeiro Costa)
Scramble: `L' U2 D' L' U2 B' D B' L U2 F2 R2 F' R2 L2 F' U2 D2 F`
`F D' + U2 * R` //EO `D' F' L2` //Pseudo 2x2x2 `F2 D F2` //Pseudo F2L-2 `B' D B` //Pseudo F2L-1 `F' D' F' L2` //All But 4 Corners `* = U R D' R' U' R D R'` //First commutator `+ = L' U' R' U L U' R U` //Second commutator 
Solution: `F D' L' U' R' U L U' R2 D' R' U' R F' L2 F2 D F2 B' D B F' D' F' L2` (25)
See on alg.cubing.net

A further comment on the solve. Notice that the commutator inserted earlier in the solve (the one labelled with +) is actually the one one found for last, even if just by one move. Vice versa the second one in the solve was inserted earlier. That’s perfectly fine!

Moreover, the two commutator cancel some move with one another.

2.4.5 Multiple Insertions: 5 Corners

Among all the cases with 5 corners left, the only one that requires 2 commutators is the one where the pieces make a 5-cycle. All other cases require 3 commutators, except when you have 5 corners placed but twisted: in that case you need 4. Here I will ignore any case requiring 3 or more commutators (although in some cases you may want to go for 3 insertions) and only look at the first one.

The easiest and probably the most used way to deal with this situation is a **two-pass** way, as for 4 corners. After having numbered the corners 1 to 5, you go through the skeleton move by

[31] Always keep in mind that when talking about commutators there is a difference between “sticker” and “piece”.

move and look for any commutator that solves a three consecutive corners cycle. These cycles are:

$$1 \to 2 \to 3 \to 1$$
$$2 \to 3 \to 4 \to 2$$
$$3 \to 4 \to 5 \to 3$$
$$4 \to 5 \to 1 \to 4$$
$$5 \to 1 \to 2 \to 5$$

This will obviously take longer than looking for just one corner cycle. It is still enough to just look for "pure" commutators. Every time you find a good commutator write it down somewhere.

Once you are done with this first pass, choose the best commutator you have found (the one cancelling the most moves)1. Insert that commutator; now you have a new skeleton that only leaves a corner 3-cycle: you know what to do.

The best choice may not be unique (for example, if you have found two different commutators canceling 3 moves). Which one should we choose? If you are short on time, just randomly choose any of them. If you have some time left, you can try both insertions and which one leads to the best second commutator.

With this method **you can't be sure** you have found the optimal solution: to be safer, you should check all (or at lest most) of the commutators you have found in the first pass and do many other (probably useless) passes. This usually leads to a huge waste of time and rarely to a better solution.

If you don't know whether you can improve your solution, keep in mind that 10 or 11 moves total are a good goal for a corner 5-cycle.

There is also a faster, but slightly more complex, way that only requires **one-pass**. It is almost identical to the first pass of the two-pass way, but besides taking note of any commutator you have found, you should also write down which corner cycle it solves. At this point, without even touching the cube anymore, you are already able to choose a pair of commutators that will solve the corner 5-cycle.

To understand how, we need some permutation theory; don't worry, I will cut it down to the essential.[32]

First of all, written with the usual notation, our cycle is:

(1 2 3 4 5)

which means that the corner 1 belongs to 2, 2 to 3,. . . and 5 to 1. With this notation, this cycle is equivalent to (2 3 4 5 1), (3 4 5 1 2) and so on. In writing down our 3-cycles (a b c) we will use the rule that our 3 digits must be consecutive in the string 1 2 3 4 5 1 2.

What we want to do is to break this 5-cycle into two 3-cycles, for example:

(1 2 3) (3 4 5)

But this decomposition doesn't work. If you want to know why you can read some of the posts I have linked in the footnote. Or, if you prefer a more practical approach. . . try it out! The important thing is that the correct possible decompositions are:

[32]An interesting further reading is the discussion started by this post [1] in the FMC thread on speedsolving.com. In particular this post [2], a mathematical proof of the "rule" for finding a 3-cycle combination to solve a 5-cycle; this other one[3], which is the same explanation I give here (it is where I have learned this technique); and this [4] is an example solve using this method.

[1]: `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-50#post-666185`
[2]: `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-50#post-666199`
[3]: `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-50#post-666209`
[4]: `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-51#post-666313`

(1 2 3) (4 5 1)
(2 3 4) (5 1 2)
(3 4 5) (1 2 3)
(4 5 1) (2 3 4)
(5 1 2) (3 4 5)

So, **the first number of the first cycle must be the same as the last number of the second cycle**.

Notice that **the order is important**: as you can see, (1 2 3) can be followed by (4 5 1) or preceded by (3 4 5). This means that once you have found a good commutator that solves, for example, (1 2 3), to complete the solve you can choose between a commutator that solves (4 5 1) somewhere *after it* or for a commutator that solves (3 4 5) somewhere *before it.* You can do the same with any of the listed pairs of cycles.

This method, although faster than the other one, doesn't allow you to check for "insertions inside insertions". Therefore, I advise using it only if you are short on time, or as "preliminary analysis": try to see how many moves you can get this way, and in the second pass only check for commutators that may lead to something better.

2.4.6 Multiple Insertions: 5 Edges

As usual, even for 5-cycles the edge case is preferably to avoid. If you decided to go that way, you can use the same method described for corners. But there is one case that can be solve in very few moves: when you can use the 6 moves 5 cycle

```
M' U M U'
```

even with some setup move, it can be really nice. Try to learn which cases it solves and possibly also some variations, like the "shifted" `L F L' R U' R'`. If you get a skeleton that leaves an edge 5-cycle, it is a good idea to number the stickers and quickly go through the solve to see if you can insert one of these algorithms. But don't waste to much time looking for it.

For more about edge insertions, see Section 4.2 about corner-first methods.

2.4.7 Other Insertions: 2 Corners and 2 Edges

Sometimes you may find a skeleton leaving 2 corners and 2 edges in a double swap (i.e.: a PLL such as J, T, V and so on).

In these cases, it is very useful to know a few **10 moves** algorithms:

`Fw2 R D R' Fw2 R D' R D R2`	(J perm)
`Rw' U Rw' U2 R B' R' U2 Rw2 B'`	(T-perm + corner twist)

There are also many 11 moves ones; some of them are:

`R U2 R' U' R U2 L' U R' U' L`	(J perm)
`R2 D B2 D' Fw2 D B2 D' Fw2 R2 U'`	(J perm)
`R2 Uw R2 Uw' R2 F2 Uw' F2 Uw F2 U'`	(T perm)
`R' U R U2 L' R' U R U' L U2`	(J perm + corner twist)

Besides all of these algorithms (this is not a complete list anyway), also their inverses and "shifted" versions solve a 2 corners - 2 edges double swap.

Note that **the inverses of these algorithms solve exactly the same case**. Just by noting this you double your chances of cancellations without the need to learn more algorithms.

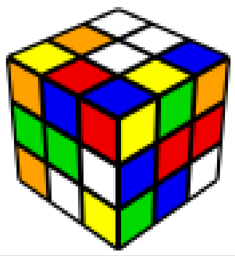
Don't expect lots of cancellations, but the more algorithms you know, the better.

As an example, see the following solve.

2C2E Insertion - Example

Scramble: `B' L' D2 R U F' U' L U2 D R2 U2 F B R2 B U2 B L2 B2 U2`

```
B' F D2    //Pseudo 2x2x1 (3/3)
L' B * R2    //Pseudo 3x2x2 (3/6)
F2 D F' D2 F    //Found using NISS (5/11)
R' D' R D2 F U2    //Found using NISS (6/17)
* = B2 L B L' B D2 F' R F D2    //2c2e insertion (9/26)
```



Solution: `B' F D2 L' R2 B R B' R2 F R' B R F' R2 F2 D F' D2 F R' D' R D2 F U2` (26)

See on alg.cubing.net

The end of the skeleton was found using NISS, so it will be a bit mysterious until you get to that section. Notice that in the same spot marked by `*` you can insert the sub-optimal J-perm `B' R2 B R B' R2 F R' B R F'`, cancelling 2 moves instead of 1 and getting the same final result.

2.4.8 Other Insertions: 3 Edges and Some Corners

In some cases you can get short skeletons (say 13 moves) that leave a 3-cycle of edges and 4 or 5 corners (or even more). Of course you can solve this case by inserting and edge 3-cycle and whatever number o corner 3-cycles is needed.

But there is another way: notice that the "sexy move" `R U R' U'` solves a 3-cycle of edges and a skew doubl 2-cycle of corners. By inserting this short algorithm and its variations you can solve the edge 3-cycle in a very effcient way. Of course the case of also geting the corners completely solved with one insertion is a very lucky accident. Often the best you can hope for is to affect the corners in such a way that you are left with one of the "good" 4 or 5 corners cases.

The "sexy move" isn't the only algorithm that can help in this cases: also block commutators are a very good tool. In this post[33] on speedsolving.com Cale Schoon gives 3 different examples of this kind of insertion. All of his solutions use NISS, but you can ignore the intermediate steps that produce the skeleton and just look at the insertions.

Another approach to this *reverse NISS* (Section 3.3), which can help you understand what you are actually doing when you insert a sexy move.

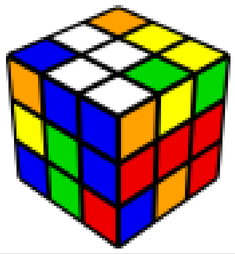
2.4.9 Other Insertions: Conjugate and Solve

A particular situation you can solve with only one insertion is when you have 4 edges and 4 corners left and both sets make a 4-cycle.

You can solve this case as follows: place all your 8 unsolved pieces on one layer (setup), so that a single move of that layer solves both the 4-cycles, perform that move (solve) and then put the 8 pieces back (anti-setup). The same technique works when the 8 pieces make four 2-cycles in total: in this case, the "interchange" will be a 180° one (for example `U2`). If the pieces are not exactly 8, or do not form the right cycles, you can again use reverse NISS after the setup moves, as explained in Section 3.3.

Let's see an example.

[33] `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-214#post-1247800`

Conjugate and Solve - Example
Scramble: `R2 L2 D2 F2 D' R2 U' B2 D' F2 U2 F' D2 L' F U B F2 U2 F2 L`
`U2 F B' L2 D2` //Two 2x2x1s (5/5) `F' * R F2 R' L2 B` //All but 4 edges and 4 corners (6/11) `* = U + L' B L' D L B' L # U'` //4 edges (9/20) `+ = F2 L' B2 L F2 L' B2 L` //3 corners (5/25) `# = L' D L U L' D' L U'` //3 corners (5/30) 
Solution: `U2 F B' L2 D2 F' U F2 L' B2 L F2 L' B' L' D L B' D L U L' D' L U2 R F2 R' L2 B` (30)
See on alg.cubing.net

As you can see, the insertion in question doesn't actually solve both the 4 cycles and the corners are completed with "normal" insertions. However, as suggested by Mirek Goljan, we could have solved everything with only one insertion, as explained above. To do so, insert in the same skeleton, at *:

```
(B D R2 B R'B2 D U2 F') U (F U2 D' B2 R B' R2 D' B')
```

This longer insertion produces the same result (30 moves).

There are also some Last Layer algorithms that work in this way. One of them is:

```
(R B2 R2 U2 R) B (R' U2 R2 B2 R')
```

and you can find some more here[34].

2.4.10 Move Count (an Estimate)

Here I will give an estimate of how many moves are usually needed for the most common types of insertions. It is a heuristic estimate, not a mathematical proven one, and keep in mind that these number also depend on:

- How many commutators/algorithms you know and your ability to recognize them.
- The skeleton's length: longer skeletons give more spots where you can insert an algorithm, and so better chance of cancellations (but you shouldn't choose a long skeleton instead of a short one because of this!).

These estimates are useful if you want to know whether it is worth or not to spend some time looking for insertions: if you goal is to achieve a solution shorter than 30 moves, if you have a skeleton leaving 3 corners in 23 moves you will probably get it, if your skeleton is 25 moves long you will need some luck.

You can also compare different kind of skeletons: a skeleton leaving 4 corner in 18 moves is probably better than one leaving 3 corners in 25.

So here are the numbers[35]

Type of insertion	**Moves**
Corner 3-cycle	5/6
Edge 3-cycle[36]	7
2 Twisted Corners (2 comms)	8
3 Twisted Corners (2 comms)	9
4 Corners (2 comms)	10
Corner 5-cycle	10/11
2 Corners and 2 Edges (double 2-cycle)	10

[34] `http://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-28#post-488378`

[35] Mostly taken from here [1], slightly adjusted to match my personal opinion.
`http://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-42#post-614593`

2.4.11 Insertion Finder

Insertion Finder[37], developed by Baiqiang Dong, is a useful tool to find insertions and check if you have failed to see something in your solutions: it finds up to 4 insertions to solve a skeleton.

It is especially useful for easy cases (3 corners or 3 edges) but in complex situation it may find solution that are not possible (or very difficult) to find for humans: use it responsibly!

2.5 Other Simple Stragies

2.5.1 Go Back and Change Your Sove

If you get stuck after a good start, you can try this: go through your solve move by move, looking for a spot where there is at least one layer containing only pieces that you have not solved yet. When you find it, you can move that layer (this is not going to affect any of the blocks you have already built). You have 3 choices (for example `U`, `U2`, `U`) and this way you will get 3 different starts that are just slightly (one move) longer than the previous. One move can be a good price to pay for a better continuation!

Of course, if you find a spot where there are two “free” layers, you can use both of them. The moves can, but don’t have to, be random: if you can see a pair forming or the EO gettig better with some moves, good for you, but sometimes you might as well try random moves and see what happens later.

2.5.2 Get Lucky!

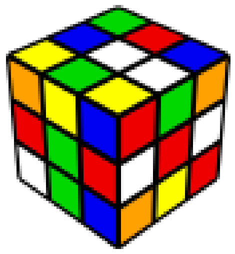
Obviously, luck is not a skill to be learned, but remember that in FMC you have to **go for it**: a “simple” solve ending with an LL skip is not less worthy than a complex unlucky one, if they have the same length. This is one of the reasons why you need to try as many different alternatives as you can: you are more likely to get a skip if you try 100 solutions than if you try 10 or 20.

2.5.3 First Example: Insert Last Pair(s)

After completing an F2L-1, you can finish the F2L by inserting the last pair. This isn’t usually a good way to continue, unless you get lucky. To improve your chances to get lucky, try to **insert the last pair in all the ways you can think of**.

For example, if your last pair is already built, you can insert it in at least 3 different ways: `U R U' R'`, `U2 R U2 R'` and `R' F R F'`. Knowing some VHF2L or ZBF2L algs can be useful to improve your chances of skip, but rather than memorizing them you should learn how they work.

As an example of alternative ways to insert pairs, consider this solve (scramble adapted from one of the German Forum Competition to avoid using premoves).

Insert Last Pairs - Example
Scramble: `D' R' U' F U2 F2 L2 D' B2 F2 D' L R' D F' U' R' B' R2 F D U' B' R' U' F`
`U2 F' U'` //2x2x2 `B' D B` //Orient two edges `D2 L D2 B2` //2x2x3 + 6 pairs `B'` //Save one pair (F2L on R) `L F L' F'` //Insert other pair `B D L' D'` //Insert saved pair `U' L2 U L U' L U L'` //Last Layer 
Solution: `U2 F' U' B' D B D2 L D2 B L F L' F' B D L' D' U' L2 U L U' L U L'`
See on alg.cubing.net

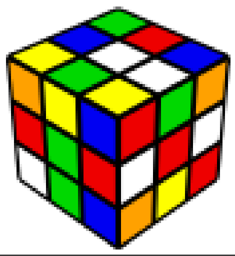
[37]`https://fewestmov.es/cube/if.cube?lang=en`

Note how I have choosen this way to insert the pairs because in the end I got a PLL skip. There was actually a better way to insert the last two pairs, which I have found using Cube Explorer (a PC software).

Insert Last Pairs - Example

Scramble: `D' R' U' F U2 F2 L2 D' B2 F2 D' L R' D F' U' R' B' R2 F D U' B' R' U' F`

```
U2 F' U'    //2x2x2
B' D B   //Orient two edges
D2 L D2 B2    //2x2x3 + 6 pairs
B' U'    //Save one pair (F2L on R)
L F L' F'    //Insert other pair
L U L B   //Insert saved pair and skip
```



Solution: `U2 F' U' B' D B D2 L D2 B U' L F L' F' L U L B`

See on alg.cubing.net

2.5.4 Second Example: How to Use Algorithms

First of all, you need to be able to recognize **symmetries** in algorithms (more precisely, in *cases*): the OLL that is solved by `F R U R' U' F'` is symmetrical about the `S` plane, so the same case can be solved by `B' R' U' R B U`. If you want to use it, try also its symmetrical to double your chances of getting a skip (or a good case). This trick works exactly because the two algorithms solve the same OLL the case, but affect the permutation of pieces in a different way. Notice however that they solve the same CLL case, so if the corners are not solved by one, they aren't solved by the other either.

An extreme example is the OLL solvable with `R U2 R' U' R U R' U' R U' R'`: with this algorithm and its symmetrical `L' U2 L U L' U' L U L' U L` you can solve the same case from 4 different angles, and from other 4 if you use their inverses. These algorithms, as all "2-gen" last layer algorithms (i.e. algorithms that only move 2 layers), doesn't affect the relative permutation of corners.

Secondly, you don't need to use an algorithm for the purpose you have learned it for. Sticking to `F R U R' U'F'`, you probably know it as an OLL. But you can decide to use it *ignoring corners*: if you complete the F2L and are left with 2 bad edges, you can try this algorithm from 4 different angles to see if you get a skip, or at least an easy case.

Chapter 3

Advanced Tools

In the previous chapter we have looked at the basic techniques, needed to find a good solution. In this one more advanced tool will be provided. They are not necessary, but they help getting unstuck and give you more possibilities to explore.

3.1 Inverse Scramble

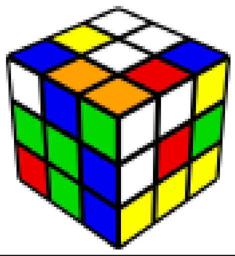
If you can't find any good start, you can try with inverse scramble: if you find a solution for the inverse sequence[1], you just need to invert it to get a solution for the normal scramble. It looks complicated but it is actually very simple.

As an example, here is the former North American record by Tim Reynolds.

Inverse Scramble - Example

Scramble: `D2 L2 B R2 U2 F' L2 U2 B2 L2 F' D L2 B U L' U2 L' F' R'`
Inverse Scramble: `R F L U2 L U' B' L2 D' F L2 B2 U2 L2 F U2 R2 B' L2 D2`

On Inverse Scramble:
`R' U F' L2` //2x2x2 (4/4)
`F2 D' B' * D2 B` //2x2x3 (5/9)
`R2 F R2 F' R2` //F2L-1 (5/14)
`F D' F' D` //All but 3 corners (4/18)
`* = B' U2 B D B' U2 B D'` //Last 3 corners (6/24)



Solution: `D' F D F' R2 F R2 F' R2 B' D' B' U2 B D' B' U2 B2 D F2 L2 F U' R` (24)
See on alg.cubing.net

To follow the solution, apply the inverse scramble first, and then the solution steps. The picture represents the "normal" scramble, not the inverse one, so your scrambled cube shouldn't match it. In the end you find the solution

```
R' U F' L2 F2 D' B2 U2 B D B' U2 B D B R2 F R2 F' R2 F D' F' D
```

for the inverse scramble, that inverted gives the actual solution written above.

It is a common mistake to think that normal and inverse scramble are complete unrelated. They are actually very similar: for example, if you use ZZ, you will notice that, for any orientation, the two scrambles have the same number of "bad" edges, but in different positions. You will also notice that any block found in one of the two is also in the other one, but sometimes it is somewhere else and made of pieces of different colors. The general rule is this:

[1]I prefer repeating myself: the inverse sequence of, for example, `F R U'` is `U R' F'`, not `F' R' U` or `U' R F`!

If in the normal scramble piece X is in the place of piece Y, in the inverse scramble piece Y is in the place of piece X.

Therefore, solved and flipped-in-place pieces will be respectively solved and flipped-in-place in the inverse scramble, with twisted corners twisted the other way round. “Fixed”[2] blocks will stay the same, while “moving” blocks will be made of different pieces and placed somewhere else.

During an official or unofficial attempt, some people write down the inverse scramble in full before starting the attempt, or when they decide to use it (or to use NISS). This way it’s easier to apply the inverse scramble any time you want without additional effort, but it takes to write it down at first (and to check for mistakes!). Instead, to apply the inverse scramble I simply invert each move in my head while reading the normal scramble from right to left. This may seem difficult at first, but in the end it becomes about as fast as aplying a regular scramble and almost without additional effort. It’s up to you to decide which method you prefer.

Besides being a technique useful as it is, in case you get stuck right at the beginning of a solve or simply to get more possibilities to explore, the ideas explained here are fundamental to the techniques introduced in the next paragraph.

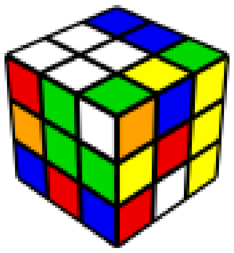
3.2 Pseudo Blocks, Premoves and NISS

This whole Section 3.2 is better read all at the same time, since the three techniques explained are deeply related.

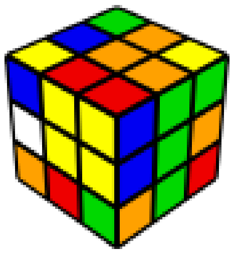
3.2.1 Pseudo Blocks

We have already met some examples of pseudo blocks. To make the concept more clear, lets look in detail at the following scramble.

Scramble: `F' L2 F2 U2 R2 B R2 F' R2 D2 U2 L' U' B' U R U L2 F2 L'`



The moves `R2 F` make a 2x2x1 block. It would be nice to expand it to a 2x2x2 in 2 or 3 moves, but the 4 moves required (`L' U B' D`) are maybe too many. But try with `L2 D'`:



View at the DFL corner of the cube after moves `R2 F L2 D'`

[2]“Fixed” blocks are those than can’t more around centers, such as a 2x2x2 or a 2x2x3. “Moving” blocks are, for example, 3x2x1s, 2x2x1s and corner/edge pairs.

What you get is not an actual 2x2x2 block, but a *pseudo* 2x2x2 block. We can think of the D layer as it was temporarily off by a D2 move, that we can do at the end to solve everything back. For example, we can continue with a (inefficient) CFOP solve:

```
R2 F L2 D'    //Pseudo 2x2x2
B' U2 R' U2 R2 U R    //Cross and second pair
U2 F' U F U' F' U' F    //Third pair
L U2 L'    //Fourth pair
B L' B' U' B U L U' B'    //OLL
F2 D' L2 D F2 R2 D B2 D' R2    //PLL
D2    //Undo premove
```

In this case the D2 move could have been done before the OLL, or between OLL and PLL, but if you don't finish the F2L as an intermediate step you have to do it at the end.

3.2.2 Premoves

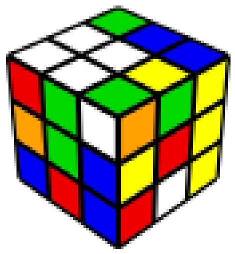
The situation of the previous example is not particularly difficult, but the "pseudoness" makes it harder to go on with the solve: the recognition of F2L pairs is not immediate. Usually it's even worse: imagine recognizing OLL and PLL when the F2L is off by, for example, R!

Someone advises to try and solve with pseudo blocks, but there is a trick that makes everything easier: you just need to perform the move that should be done at the end (in this case, D2) **before the scramble** (and that's why they are called "pre-moves"). Try it out!

Premoves - Modified Scramble Example

Scramble: *D2* `F' L2 F2 U2 R2 B R2 F' R2 D2 U2 L' U' B' U R U L2 F2 L'`

```
R2 F L2 D'    //2x2x2
B' U2 R' U2 R2 U R    //Cross and second pair
U2 F' U F U' F' U' F    //Third pair
L U2 L'    //Fourth pair
B L' B' U' B U L U' B'    //OLL
F2 D' L2 D F2 R2 D B2 D' R2    //PLL
```



See on alg.cubing.net

Once you have found such a solution, remember that the premove (or premoves, if there is more than one) has to be **added at the end** of the solution, to solve the original scramble. In this way we get back the final solution of the previous sub-section.

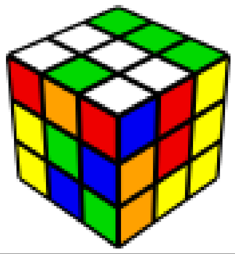
You can use **more than one premove**: take for example this solve, which is my first official solve as well as former Italian National Record. Remember to permorm the premoves before starting scrambling.

Multiple Premoves - Example

Scramble: `U L' F' L2 F' D2 F'B' U' R2 U L' F2 U' F2 L2 U2 F2 L2 U2`

Premoves: D2 B2

```
R2 B' R2 B    //2x2x2, found premove B2 here
D L2 F D F2    //2x2x3
L' D F' D2 F D'    //F2L-1, found premove D2 here
L' D * L' F L' F' D' L'    //All but 3 corners
* = B L' F L B' L' F' L    //Last 3 corners
```



Solution: `R2 B' R2 B D L2 F D F2 L' D F' D2 F D' L' D B L' F L B' L2 F' D' L' D2 B2` (28)

See on alg.cubing.net

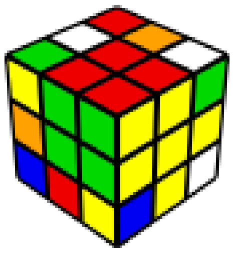
3.2.3 NISS

In the last example, I have found the two premoves in two different moments. It is a little harder to recognize pseudo blocks that require more than one premove: with our scramble

`F' L2 F2 U2 R2 B R2 F' R2 D2 U2 L' U' B' U R U L2 F2 L'`

start with the same 2x2x1 (`R2 F'`). Even an expert will find it difficult to see that by adding `D F'` as premoves you get a 2x2x2:

Scramble: `D F' F' L2 F2 U2 R2 B R2 F' R2 D2 U2 L' U' B' U R U L2 F2 L'`
2x2x1: `R2 F`



2×2×2 in DFR after `R2 F`, with premoves `D F'`.

But such premoves are not har to find, if you know **NISS** (Normal-Inverse Scramble Switch). This technique was first introduced by Guus Razoux Schultz in 2009 and we explanation we outline here is taken form this excellent post[3] by Tomoaki Okayama. The most important fact is the following:

The scramble and the solution can be thought of as a single move sequence loop, that doesn't affect the cube in any way.

For example, let abstractly `A B C D` be a scramble and `p q r s` a solution for it. The sequence `A B C D p q r s` brings the cube back to it solved state. In the same way, every "shifted" version of this sequence:

`s (A B C D p q r s) s'` = `s A B C D p q r`
`r s (A B C D p q r s) s' r'` = `r s A B C D`
`q r s (A B C D p q r s) s' r' q'` = `q r s A B C D`
`p q r s (A B C D p q r s) s' r' q' p'` = `p q r s A B C D`
`D p q r s (A B C D p q r s) s' r' q' p' D'` = `D p q r s A B C`
...

doesn't affect the cube. Inverse sequences don't affect the cube either, obviously.

In our first example with premove `D2`, the loop would have been:

(Scramble) `R2 F L2 D'` (Other moves) `D2`

And preforming `D2` at the beginning only means taking one of the shifted variations:

`D2` (Scramble) `R2 F L2 D'` (Other moves)

In other words, we can consider "`R2 F L2 D'` (Other moves) `D2`" as a solution for "(Scramble)" or "`R2 F L2 D'` (Other moves)" as a solution for "`D2` (Scramble)"[4].

This should be enough to understand how premoves work. Knowing this, we can also find a solution partly on the normal scramble scramble and partly on the inverse one. How? Consider the same scramble and the same start `R2 F`. We know that, starting from here, our solution will look like `R2 F (W)`, where `(W)` stays for some sequence of moves. Our loop sequence is:

[3] `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-52#post-667292`

[4] You can also see the scramble as a solution to the solution!

(Scramble) `R2 F (W)`

As we said before, the inverse of this is also an “identity” loop (It is equivalent to finding a solution using the inverse scramble):

`(W)' F' R2` (Inverse Scramble)

We can therefore consider “`(W)' F' R2`” as a solution for “(Inverse Scramble)” but also, for example, `(W)'` as a solutin for “`F' R2` (Inverse Scramble)”. In this way we have come to the general rule

You can use the inverse of the moves found on normal scramble as premoves for the inverse scramble.

Suppose now you have found a solution, call it `(K)`, to the inverse scramble with premoves `F' R2`. Then `(K)` must have the same effect as `(W)'`, so you are done: your final solution would be `R2 F (K)'`.

You can repeat this process: suppose you have found the moves `F D'` on inverse scramble with premoves (which make a 2x2x2 block), but no good continuation. At this point we can go back to normal scramble, using `D F'` as premoves. In fact, the loop sequence is:

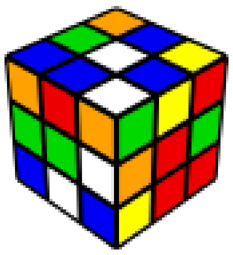
`F' R2` (Inverse Scramble) `F D'` (Moves yet to be found)

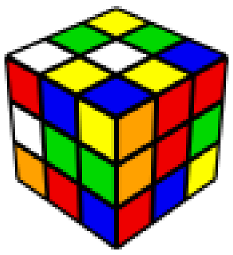
Inverting it gives another identity loop:

(Inverse of moves yet to be found) `D F'` (Scramble) `R2 F`

So we can consider `D F'` as premoves for the original scramble and start with `R2 F`.
An example[5] will make everything clearer.

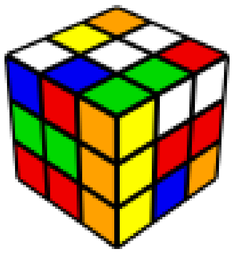
Scramble: `R B U L' U' B U2 D2 F D R L D B2 L2 D' F2 D2 L2 D`
Inverse Scramble: `D' L2 D2 F2 D L2 B2 D' L' R' D' F' D2 U2 B' U L U' B' R'`


Normal Scramble


Inverse Scramble

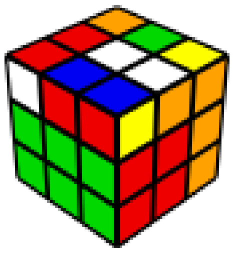
Explanation:

Nice start on inverse scramble: `B L' U F2`


(Inverse Scramble) `B L' U F2`

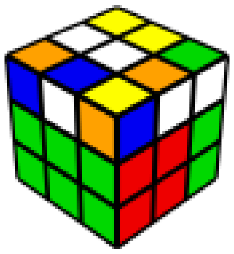
[5]Taken from here: `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-10#post-258791`.

Apply on normal scramble: [pre-moves `F2 U' L B'`] nice continuation: `D' B' U B2 R2`



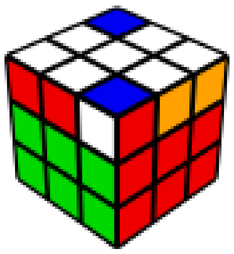
F2 U' L B' (Scramble) D' B' U B2 R2

Apply on inverse scramble: [pre-moves `R2 B2 U' B D`]: `B L' U F2`



R2 B2 U' B D (Inverse Scramble) B L' U F2

Easy continuation on inverse scramble:
F2L: `R B R' U R' U2 R B`



R2 B2 U' B D (Inverse Scramble) B L' U F2 R B R' U R' U2 R B

LL: `U2 R' U2 R' D' L F2 L' D R2`
Premoves correction: `R2 B2 U' B D`
Solution: `D' B' U B2 D' L F2 L' D R U2 R U2 B' R' U2 R U' R B' R' F2 U' L B'`

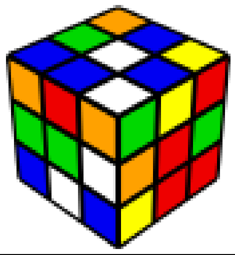
To make writing such solutions shorter and easier, I have proposed the following notation: put moves that are done on inverse scramble are written inside round brackets. This notation is now widely accepted and used among FMC solvers, but can still confuse beginners who don't know NISS if you use it without explaining it.

For example, Guus' solve becomes:

NISS - Example

Scramble: `R B U L' U' B U2 D2 F D R L D B2 L2 D' F2 D2 L2 D`
Inverse Scramble: `D' L2 D2 F2 D L2 B2 D' L' R' D' F' D2 U2 B' U L U' B' R'`

```
(B L' U F2)   //Nice start on inverse scramble
D' B' U B2 R2   //Nice continuation
(R B R' U R' U2 R B)   //F2L
(U2 R' U2 R' D' L F2 L' D R2)   //LL
```



Solution: `D' B' U B2 D' L F2 L' D R U2 R U2 B' R' U2 R U' R B' R' F2 U' L B'` (25)

When you have to write down the final solution, just go though all the moves done on the normal scramble and then backwards through the moves on inverse scramble, inverting each of them. For example, if you have written down the solution with the brackets notation

```
A B
(C)
D
(E F)
(G)
```

the final solution is

```
A B D G' F' E' C'
```

3.3 Reverse NISS

It is not a widely used technique, but it can occasionally be useful: it can be considered an improvement over both "Conjugate and Solve" and "Go Back and Change your Solve".

Suppose you have found a good skeleton solving everything but a few pieces (from 4 to 8). You would like to insert an algorithm to solve them, but if these pieces don't have at least one color in common (they don't belong to the same layer) it can be hard to recognize which algorithm to use. The trick is this: if you find a spot in the solve where all your unsolved pieces are conjugated to one layer, you can **"split" the solve there**, using all the following moves as premoves (that is why I called it "Reverse NISS"); at this point, you have only a "Last Layer" left to solve. If needed, you can use some setup moves.

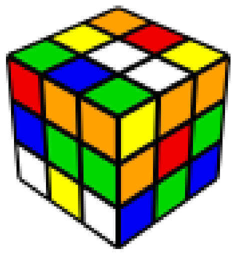
Let's take this solve as an example[6].

Reverse NISS - Example

Scramble: `L2 D2 F2 U' L2 D L2 D2 B2 U' F' U' R' D B2 L D B' F' R'`

Premove: R2

```
F2 U R' U' F D2 * F' U2   //2x2x3
R D R' D2 B2   //F2L (All but 5 pieces)
R2   //Undo premove
* = (F R) F D F2 R F R2 D R D2 (R' F')   //5 pieces
```



Solution: `F2 U R' U' F D2 F R F D F2 R F R2 D R D2 R' F2 U2 R D R' D2 B2 R2` (26)
See on alg.cubing.net

[6]Actually, in this solve, the algorithm was easy to recognize simply by marking the pieces with arrows and Xs, so I didn't use this technique, but this solution is a good example anyway.

The moves `F R` in the insertion conjugate all the unsolved pieces to the same layer. If you add them the skeleton becomes

```
F2 U R' U' F D2 F R * R' F' F' U2 R D R' D2 B2 R2
```

and we can "split" it at `*`: use `R' F' F' U2 R D R' D2 B2 R2` as premoves for the normal scramble and you have:

Premoves: `R' F' F' U2 R D R' D2 B2 R2`
F2L: `F2 U R' U' F D2 F R`
LL: `F D F2 R F R2 D R D2`

3.4 Useful Algorithms

As you can see, in the last example solve I have used an OLL that is maybe not well known, that is `R U R2 F R F2 U F (U2)` (modulo rotations). This one in particular is very useful, because it is the shortest algorithm that affects the orientation but not the permutation of pieces.

It is in general useful to know some of the shortest last layer algorithms, **up to 9 or 10 moves**. You can find a complete list (modulo rotations, symmetries and inverses) in Appendix C.

If you decide to break the "never build an F2L without influencing the last layer" rule (sometimes it is worth trying!) you can hope the last layer can be solved with a short algorithm: in this case, the more you know, the better!

A little tip: when using a last layer algorithm, remember that you can try performing the AUF ("Adjust Upper Face") both before and after it, hoping to cancel some move with the ones before it of with premoves.

There are two other reasons why it is worth learning some algorithm, at least the ones from 6 to 9 moves. The first one is that even if you don't know the exact algorithm for the last layer, or if you haven't completed the F2L, one of these algorithms may leave you with a good skeleton (for example, a corner 3-cycle), hopefully canceling some move.

The second reason is that by studying algorithms you can learn to match blocks or complete the F2L. Let's take a look at the optimal T perm:

```
R2 Uw R2 Uw' R2 y L2 Uw' L2 Uw L2 (U')
```

It is just a useful F2L algorithm repeated twice.

Besides just learning algorithms, you can also **learn *from* the algorithms**.

3.5 Pair Analysis

This is a really obscure technique, based on intuition, not proven to actually give you some advantage during the solve. What is it about? "Analyzing pairs" means taking into account some things about the scramble, which are:

- Ready made pairs. There isn't much to say.
- Pairs that can be made **with only one move**. To find them you can use the "brute force" technique described earlier or try to recognize them at first sight. Moreover, it can be useful knowing how to recognize **pseudo pairs**, that is pairs that can be solved with only one move on inverse scramble. Moves that make a pair, both on normal and on inverse scramble, can be taken as first move, or as a premove for the inverse of the scramble you are considering.
- **Bad Pairs**: those corner/edge pairs that are wrongly matched, because one of the pieces is misoriented. Intuitively, such pairs are bad and you want to break most of them as soon as you can.

There isn't much documentation about this technique, especially for bad pairs. Guus Razoux Schultz did a good analysis for the first scramble[7] of Twente Open 2012 in this post[8] on speedsolving.com.

3.6 Solving with Skew Centers

This technique is understood more naturally in the context of corners first solving, but can be used with any method.

We have seen insertion for corners and edges, but we can also solve centers using insertions: if you ignore centers during the solve and leave them unsolved, as long as they are off by an even number of slice moves[9], you can solve them with algorithms such as `M E M' E'` or `M E2 M' E2`[10].

Of course, since the moves required are all slices (or "inner layer" moves), the number of moves required to solve the skew centers is 8. But don't worry: there are many, many chances to cancel moves. I estimate that, with insertions, you only need 3 moves on average to solve centers. How is this possible? Notice that there is more than one way to solve them:

`M E M' E'` = `S M S' M'` = `E S' E' S`

and

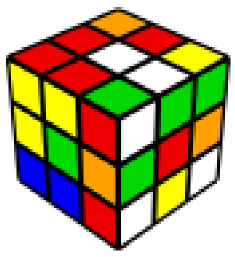
`M E2 M' E2` = `M' E2 M E2` = `M S2 M' S2` = `M' S2 M S2`

and in this second case, all inverse sequences solve the same case!

Sticking to the first case, as soon as you have a slice move in your (partial) solution, you know you can cancel at least 4 moves: each of te 6 slice moves appears at the beginning or at the end of one of the algorithms above.

The downside of this method is that it takes a lot of rewriting: when you move centers around, the moves you have written need to be changed to affect in the same way the other pieces. As a result, it can be really time consuming if you aren't confident with it.

As an example, take my last solve from World Championship 2017 (which features also NISS and a corner insertion). Notice that, ignoring centers, a 2x2x2 block is already solved!

Skew Centers - Example
Scramble: `R' U' F U R2 B2 D' L2 F2 D U' F2 R' B D' F' U' L R D F2 U F' R' U' F`
`(U2 L2 B2 U)` //3 pairs (4/4) `R2 D2 R2` //blocks (3/7) `(L B')` //2x2x3 + square (2/9) `(L2 U2 L' U2 L' U2 L)` //All but 3 corners (7/16) Skeleton: `U2 L2 B2 U L + B' L2 * U2 L' U2 L' U2 L B2 L2 B2` `* = F' D F U2 F' D' F U2` //3c (6/22) `+ = M' E M E'` //Centers (4/26) 
Solution: `R2 D2 R2 D' B2 D B2 D L' F L B2 L' F' L D2 L B' F D U' R' U' B2 L2 U2` (26)

To find this solution, at first I did `S' M S M'`, or one of the equivalent sequences, at the beginning of the solve. Then I went on with finding a solution as I would do in a normal solve. From this point on the final solution can be derived without touching the cube anymore.

[7]https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-61#post-721325
[8]https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-62#post-721942
[9]This condition is required to avoid parity.
[10]These are actually commutators: `M E M' E'` = `[M, E]` and `M E2 M' E2` = `[M, E2]`.

After I found the solution (including the insertion), I had to rewrite all the moves as if I had done them without solving the centers first. This is a simple translation: `R` became `U`, D became `F` and so on.

Then I had to insert one of the three commutators to solve centers. Since centers don't move around in a sequence of moves without rotation or slices, the commutators that solved centers remained the same at every point. Knowing this, performing the moves on the cube is unnecessary: you just need to find a spot where one of the three commutators cancels the most.

Finally, after inserting the slice moves that solve centers, there is the second (and last) rewriting: the moves before the insertion remain unchanged, but the moves after that have to be canged with same method used the first time. If you are disciplined in writing down your partial solutions, you can get the first version of your solution and copy the last moves, but this is not my case, as I write stuff down on my sheets in a very chaotic way.

Chapter 4

Some Other Methods

The “standard” method is building a skeleton with blockbuilding and then solving the last few pieces with insertions. But there are some other approaches worth mentioning.

4.1 Starting With EO

Starting by orienting all edges, as you would do in a ZZ solve, is a possiblity to always keep in mind. Since the first version of this tutorial I have used it more and more in my solves, to the point that at the beginning of an attempt I always look for all possible EOs on normal and inverse scramble to see if they are worth a try: they often are. As mentioned in Section 2.1.7, there are many notable FMCers that often start with EO.

Remember that there are 3 possible orientation with respect to which you can orient edges: with respect to F/B (reduce to `<R, L, U, D>`), to R/L and to U/D. If you procede with a normal ZZ solve, for each of these you can build the F2L on 4 different sides. Even if you don’t like starting with EO, I suggest practicing some (color neutral!) ZZ to improve the EO recognition.

From here you have two ways to continue.

4.1.1 EO + Blockbuilding

After having oriented all edges, the most common way to go on is blockbuilding. The pro is that we don’t have any “bad” edge, but this forces[1] us not to use moves that break the EO, and this is a (relatively small) limit.

Since you have usually more than one (nice) way to orient edges for a given orientation, you should also try to build as many pairs/blocks as you can during the EO step. As an alternative approach, you can pay attention to EO while you build the first block(s) (for example, a 2x2x2) and orient edges immediately after that.

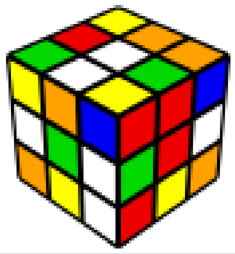
Here the solve that made Gregorz Łuczyna the 2010 European Champion. Notice that he starts by rotating the cube to his preferred orientation. This makes it easier to spot blocks if you are not used to color neutrality, but I dislike this habit. See Section 6.1 for more about how to write down a solution.

[1]Obviously, no one is forcing you to do anything, but orienting edges and then destroying what you have just done doesn’t look like a smart thing to do. You can also start with a *partial* EO if you wish.

EO first - Example 1

Scramble: `L D2 B' D2 B R' B' U B L B L2 B2 U2 F2 U R2 D' B2 D' B2`

```
x y2 L2 D F'    //EO (3/3)
R L2 D *    //EOLine (3/6)
R' U2 B2 R2 B2    //2x2x3 (5/11)
L2 U' R' U L U' L' R U2 L' U'    //All but 3 corners (11/22)
* = D2 R' U' R D2 R' U R    //3c (4/26)
```



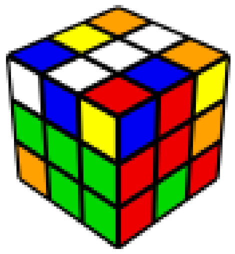
Solution: `x y2 L2 D F' R L2 D' R' U' R D2 R' U' B2 R2 B2 L2 U' R' U L U' L' R U2 L' U'` (26)
See on alg.cubing.net

Here's another example: the first solve of João Pedro Batista Ribeiro Costa at World Championship 2015, part of his 25.67 winning Mean of 3.

EO first - Example 2

Scramble: `L2 U' B2 L2 D' F2 L2 D U' L2 U2 B' R2 B' R B R' D' B' F2`

```
U2 R' U2 * R'    //EO (4/4)
B F2 U2 F    //Pseudo 2x2x3 (4/8)
(B L2)    //2x2x3 (2/10)
B2 U' B2 U' B2 U' B2 U' B' U    //All but 3 corners (10/20)
* = U R' D2 R U' R' D2 R    //3c (6/26)
```



Solution: `U2 R' U' R' D2 R U' R' D2 B F2 U2 F B2 U' B2 U' B2 U' B2 U' B' U L2 B'` (26)
See on alg.cubing.net

4.1.2 Domino Reduction (and HTA)

Edge orientation can be considered, modulo rotations, a reduction to the *subgroup* generated by the moves `<R, L, U, D>` or, equivalently, `<R, L, U, D, F2, B2>`. In other words, by orienting edges you reduce the cube to a case that can be solved using only the moves `R, L, U, D, F2` and `B2`. Another step in this direction leads to reducing the cube to the subgroup generated by `<U, D, R2, L2, F2, B2>`; to do so you have to:

- Place E layer edges on the E layer;
- orient corners.

This reduction is also called "Domino", because it makes a Rubik's Cube solvable as a 3x3x2 "cube" (also called "Domino Cube"). Moreover, these are the first two steps of Human Thistlethwaite Algorithm (HTA), a modified version of Thistlethwaite Algorithm. If you are interested in this method for FMC, I suggest this tutorial[2].

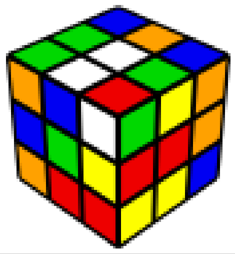
Here is an example solve by Per Kristen Fredlund.

[2]https://www.speedsolving.com/forum/threads/guide-human-thistlethwaite-algorithm-for-fewest-moves-hta-for-fm
31704/

EO first - Example 2

Scramble: `R2 F2 L2 D' R' U' R D' F B R U B2 L2 D2 F2 L2 D B2`

```
R' B U' D F     //EO (5/5)
L' F2 L     //Domino reduction (3/8)
D2 L2 F2 D F2 D L2 U' R2 D2 R2     //Finish (11/19)
```



Solution: `R' B U' D F L' F2 L D2 L2 F2 D F2 D L2 U' R2 D2 R2` (19)
See on alg.cubing.net

One last consideration: in the examples above there are two nice and **short** EO steps. But this doesn't mean you shoul discard a longer EO, if you can build a lot of blocks while doing it!

4.2 Corners First

"Corners First" (sometimes shortened to CF) is not really a method, but a class of methods that, as the name says, solve the corners first, and then the edges. Roux can be considered a CF method.

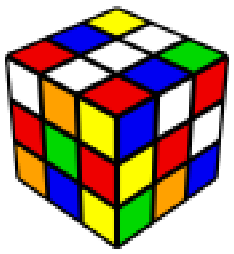
Among the ones who have figured out how to solve the cube on their own, many had a corners first approach:[3] Thinking separately about corners and edges makes it somehow easier to solve the cube intuitively. Moreover, by solving the corners first you can solve edges more freely: inner layers can be moved without affecting corners.

But this is also a disadvantage in FMC: inner layer moves count as two moves! Despite this, there at least two expert FMCer that use this technique: **Attila Horváth** and **Javier Cabezuelo Sánchez**, Spanish national record holder. Both agree that Corner First methods are excellent for FMC, but not very suitable for the one hour time limit. In fact, many of Javier's official results are DNF.

Attila Horváth mostly solves corners using a method similar to Guimond (orient first). Centers are not cared about in this step. For this step he sometimes uses premoves or NISS. After this, he goes through the solution he has found and modifies it slightly, inserting inner layer moves, to get at least 2 or 3 edges solved. At the end he solves the edges left, in no specific order. He sometimes doesn't solve the centers until the end, and solves them with an insertion, as discussed in Section 3.6. To learn more about his method, I suggest reading his posts or asking him directly on the speedsolving.com forum[4]. He is usually very happy to teach others about his techniques.

Here is a commented solve by Attila.

Scramble: `U L U' R' F' L' U D R L' B2 D B' D L2 D' R2 U F2 D`



[3]For example, Valery Morozov, who has made a tutorial to learn his method, available here: `https://www.speedsolving.com/forum/threads/a-unique-phase-method-for-a-rubiks-cube.44264/`.

[4]Here is his profile: `https://www.speedsolving.com/forum/members/attila.10652/`.

Solution: `B2 D R' B' D2 U2 F L D' U' R' U2 R' B F2 D2 L2 U' D R U' R' L B2` (24)

Explanation:

First I need a short corners-solution, usually I try something with premoves, if the scramble seems too hard. In this case I found this premoves for normal scramble: `D B`.

Corners solution:
`B2 D' B' D2 B` *(Guimond first step: orient corners)*
`B2 D' R2 F2` *(Solve all corners)*
Corners solve, without premoves:
`B2 D' B' D2 B' D' R2 F2 D B`
Corners solve for inverse scramble: (inverse of previous solve)
`B' D' F2 R2 D B D2 B D B2`

A variation of the previous solve, to get more edges solved:
`B2 M b d' M' F2 R2 d2 D' b d2 B` *corners -2 moves and 5 edges solve,*
Then I write this, without centers move:
`B2 L' R U R' D' U L2 D2 F2 B' R U2 R`
The second move (`M`*) does not change the first five edges position, but it must be inserted to get the lucky ending.*

The next step is obvious, solve more 3 edges:
`U D` *setup moves,* `L' F' U2 D2 B R` *3 edges algo, then a lucky E slice skip, due to the previous M move.*

Since the first version of this tutorial, Attila has gradually changed his method. He still orient corners first, but rather solving them completely and then taking care of edges, he reduces somehow to a domino solve. Se for example this post[5].

Javier Cabezuelo Sánchez solves the corners in a different way: first layer corners first, then the other. He then tries to solve the edges inserting moves (or algorithms) in the solution he has found. He doesn't use techniques such as inverse scramble, premoves or NISS. Differently from Attila, he cares about centers while solving corners. See also this post[6].

Both Attila and Javier only use their CF method, which breaks the "never restrict yourself" rule; but they still get excellent results.

[5]https://www.speedsolving.com/forum/threads/the-3x3x3-example-solve-thread.14345/page-280#post-1234805

[6]https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-111#post-945295

Chapter 5

How To Practice

Many people say that to get better you need to "practice, practice, practice". It is true, but you also need to know *how* to practice: here is some advice on how to practice to get better at FMC.

5.1 No Time Limit and Competition Simulation

Always trying to simulate a competition and forcing yourself to complete your solution in one hour is not the best thing to do: on the contrary, I suggest **not limiting your time** and trying the same scramble again and again until your are satisfied with your result.

This doesn't mean doing one hour solves is bad: it tells you what your level is and it helps you finding a good time management strategy[1]. To train like this I suggest taking part in online competitions such as the one hosted on David Adams' website[2] and the German Forum competition[3].

Trying for one hour as it was a competition and then keep trying until you reach a good result is a balanced compromise.

5.2 Compare Yourself to the Masters (and Study their Solves)

When you want to practice, I suggest trying a scramble that has already been solved by some expert FMCer, so that you can compare your solution to their and find out whether you have missed a good start or anything else. Steal all the secrets you can!

Moreover, to train blockbuilding and other methods it is mandatory to study the masters' solutions: you can find many of the them lookig at the old rounds of the online competitions cited before or on the FMC Thread on speedsolving.com. A couple of years ago I started a blog-style website `http://fmcsolves.cubing.net/` to collect nice FMC solves, but I haven't updated it in ages.

5.3 Hard Scrambles

To see what you can do in the "worst case scenario", I suggest trying out some scrambles that are considered by experts to be really hard. You can find a list of hard scrambles here[4].

[1] I will talk about time management in Section 6.3.
[2] `https://www.ocf.berkeley.edu/~dadams/fmc/`
[3] `https://speedcube.de/forum/showthread.php?tid=5795`
[4] `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/page-88#post-842681`

5.4 Deliberate Practice

If you think you have troubles in finding a good start, practice that: take a scramble, find a 2x2x2, 2x2x3 or something else until you are satisfied, then change scramble. You can apply this idea to F2L-1 or any other substep.

5.5 Fast Scrambling

Even if it is not necessary, when using techniques like NISS, or simply if like me you tend to solve and scramble the cube many times during a solve, you should try to be at least "not too slow" at scrambling, and most important to be accurate (don't make mistakes). 10 seconds for a 20 moves scramble is fine.

5.6 Study!

Last but not least. Study this guide, study from other sources, study algorithms and techniques, new or known. I have read "The FMC Thread" on speedsolving.com twice, from top to bottom.

Study algorithms: there are dozens of different sets, to mention some of them LLEF[5] (Last Layers Edges First) or Summer Variation[6]. Remember that you shouldn't just *memorize* them, but also try to understand *how they work*.

[5] `https://www.speedsolving.com/wiki/index.php/LLEF`

[6] `https://www.speedsolving.com/wiki/index.php/Summer_Variation`

Chapter 6

In Competition

6.1 How to Write a Solution

Both in competition and while practicing, you should write down your solution **without rotations**. There are many good reasons to do so:

- Using rotations, it is easier to make mistakes.
- Rotations can hide cancellations: things like ... `R z' U'` ... are a terrible way to waste moves!
- While solving, if you rotate the cube, you always need to keep in mind which side is where.

How to write a solution without rotations? A PLL in B is very awkward. There is an easy way: keeping the standard orientation, with a BOY color scheme (the “standard” one), you just need to remember that the white-centered layer is always `U`, the green-centered one is `F`, the yellow-centered one is `D` and so on. Every time you move a layer, for example the white-centered one, you don’t need to care about how you are looking at the cube at that moment: just write `U`.

To help memorizing the scheme (not that it is hard), remember that Blue and Red begin with the same letter as their layer. This trick actually works well in many other languages.

6.2 Backup Solution

It is good habit, in time-limited competitions, to write a “backup solution”. It is usually a not so good solution, but still better than a DNF, found before the last moments, when haste may lead to making a mistake or not finding a solution at all. If, for example, you average about 35 moves, but after 20 minutes you have found and written down somewhere a 40 moves solution, you will be more relaxed for the remaining 40 minutes. You can even write it down on the official sheet: if you later want to change your solution, you can delete the backup solution and write down the new one. There are many possible approaches to finding a backup solution:

- Force yourself to have found and written a solution, no matter how bad, in a fixed time limit (i.e.: 35 minutes). I don’t do this, but it can be useful if you often find yourself at the end of the hour without anything written down.
- If you casually come across some solution (i.e.: you have found a good start and solving the cube to re-scramble it you get a PLL skip), take note of it somewhere.
- What I do: I don’t really find backup solutions, but many backup skeletons. For example, my goal is usually sub-30; in this case, a skeleton leaving 3 corners in 26 moves is not good, but if I find one I keep it somewhere. If I have, for example, 10 minutes left and I don’t have anything better, I look for an insertion in that skeleton. A single 3c insertion usually takes me about 5 minutes, so you should adjust my “10 minutes” to your speed.

What can a good backup solution be? Any solution! Anything is better than a DNF, especially now that the preferred format for FMC (in official competitions) is "Mean of 3": a single DNF gives you a DNF mean.

6.3 Time Managment

"How to manage your time" is a complex topic, and I don't want to say that my advice is absolutely good in any case: follow it carefully! In fact, I consider myself pretty bad at time-managment. The best teacher, in this case, is personal experience.

6.3.1 Don't Get Stuck

It can happen to anyone: during a competition you get stuck on a certain start and don't seem to find any better continuation. The ability to quickly understand if a start can lead to a good continuation would be as useful as being able to read other people's mind (in the FMC world only). My advice, maybe trivial, is: don't get stuck. If you have tried every method and technique you know and found nothing, don't stare at the cube hoping it solves itself: go back and try something else.

6.3.2 (Don't) Explore Every Possibility

In the first version of this tutorial this section was called "Explore Every Possibility" - a radical change! The content of the old section is still valid though:

If you are computer scientist, mathematicians or so I can tell you that exploring different possible solution is actually a tree search[1]*: you can choose if you prefer a DFS (depth-first search, trying to complete a solution before moving to another one) or a BFS (breadth-first search, "advance" every solution in parallel) approach, or a mixed one. Keep in mind that form a single partial solution can derive a lot of nice branches as well a none; for this reason I don't use a fixed method. It is also important to know when to prune a branch (that is, discarding unpromising partial solutions).*

Why did I change the title? Simply because in the last three years I have realized that my obsession for not missing any (promising) start and/or continuation gave me a strange (and wrong!) attitude during the solves: there were moments when I tried not to find a good continuation for the most promising start(s), but to convinve myself that certain starts were not good and had to be discarded. In the computer science language used above, I was trying to hard to prune bad branches of the tree, while I should have been looking for the most fruity ones.

For example, I used to always check both the normal and inverse scramble for EO and blockbuilding starts. Now, if I find something very good immediately, I postpone or even skip checking other stuff or the inverse scramble.

After years of practice, I still don't know what is the best way to choose which partial solutions to explore and which to discard. All I can do is tell you what the problems are, so that you can try and find a way to solve them on your own.

[1] https://en.wikipedia.org/wiki/Tree_(data_structure)

Appendix A

Other Resources

Here is a list of the sources where I got all this information from and other useful links.

Speedsolving.com

The speedsolving.com forum is the place where all the knowledge come from. In particular:

- The FMC Thread: `https://www.speedsolving.com/forum/threads/the-fmc-thread.13599/`

  A thread dedicated to FMC, where people constantly post their results and ask for advice.

- Fewest Moves: Tips and Techniques: `https://www.speedsolving.com/forum/threads/fewest-moves-tips-and-techniques.1566/`

  A thread by Arnaud van Galen collecting the most useful techniques, already included in this tutorial.

- A Tutorial for Corner Commutators by Brian Yu: `https://www.speedsolving.com/forum/threads/bh-tutorial.12268/`

Other Tutorials

- A video by Daniel Sheppard with advice on how to get better at FMC: `https://www.youtube.com/watch?v=q0mrMD933rM`
- A 5-part video tutorial by Ranzha, first part: `https://www.youtube.com/watch?v=-gKAzXYonHI`
- A presentation by Pranav Maneriker: `https://prezi.com/cng_isud-im-/rubiks-cube-fewest-moves/`

Online Competitions

Online competitions are useful not only for competing and testing yourself against other people on the same scramble, but also to study multiple good solutions to the same scramble: have a look at the past rounds!

- David Adams' Online Competition: `https://www.ocf.berkeley.edu/~dadams/fmc/`
- German Forum Competition: `https://speedcube.de/forum/showthread.php?tid=5795`

Cube Solving Programs

Cube solving programs can be useful to compare your solution with the optimal one, especially for the first blocks.

- Cube Explorer, a cube solving program by Herbert Kociemba: `http://kociemba.org/cube.htm`

  It can be used, for example, to find the optimal algorithm for a given case or the best possible ending for a partial solution (see also the last example in Section 2.5.3). Pay attention in this last case: you may beat the optimal solution with insertions!

- Insertion Finder, by Baiqiang Dong: `https://fewestmov.es/cube/if.cube?lang=en`

  This tool is very useful to check if you have found optimal insertions for a given skeleton.

- HARCS, by Matt DiPalma (replacing JARCS by Johanes Laire): `https://www.speedsolving.com/forum/threads/harcs-jarcs-replacement-cube-solver.63241/`

  This tool is useful to find optimal solution to substeps of common metods (in fact, of any method).

Other Websites

- Ryan Heise's website: `http://www.ryanheise.com/cube/`
  - In particular, the *Fundamental Techniqes* section: `http://www.ryanheise.com/cube/fundamental_techniques.html`
- Lars Petrus' website, with useful blockbuilding examples: `http://lar5.com/cube/`

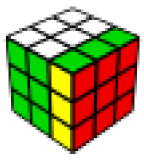
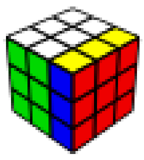
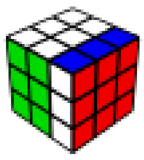
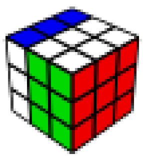
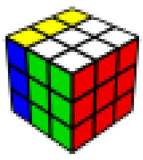
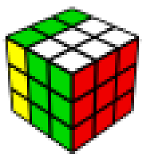
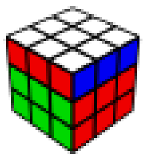
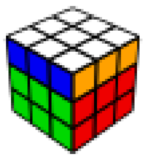
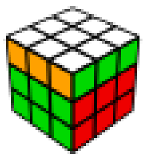
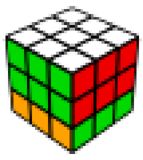
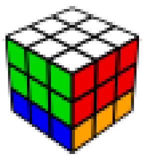
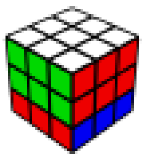
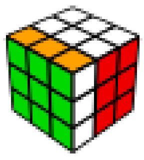
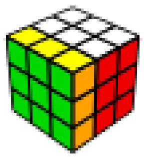
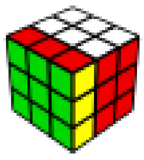
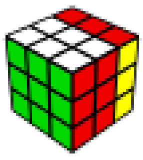
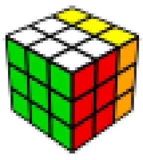
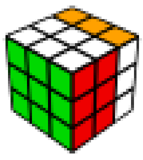
Appendix B

Notation

The standard (OBTM) notation is actually very simple to learn. The basic rules are the following:

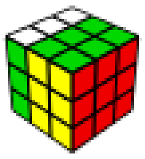
- To each face is assigned a letter: `R` = Right, `L` = Left, `U` = Up, `D` = Down, `F` = Front and `B` = Back.
- Without additional symbols, the letter means “turn that face 90 degrees clockwise”. Modification are: “2” (for example, `U2`) for 180 degrees turns; `'`(prime or apostrphe, for example: `U'`) for 90 degrees counter-clockwise turns.

In the following table you can see all the basic moves, that are the only ones you need to write down your solution. See Section 6.1.

 `R`	 `R2`	 `R'`	 `L`	 `L2`	 `L'`
 `U`	 `U2`	 `U'`	 `D`	 `D2`	 `D'`
 `F`	 `F2`	 `F'`	 `B`	 `B2`	 `B'`

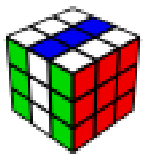
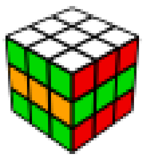
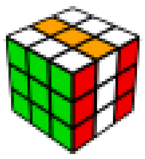
Other Moves

Other moves include:

- Wide moves: denoted by a lowercase `w` after the letter (for example `Uw` or `Rw’` or `Dw2`) denote “wide turn”: you have to move the inner layer together with the outer one. For example:  `Rw` They are sometimes also denoted by a lowercase letter (The move in the picure would be `r`), although this notation is not standard.

- Cube rotations: denoted by a lowercase letter in square brackets, or by `x`, `y` and `z` (with the rule `x` = `[r]`, `y` = `[u]`, `z` = `[f]`) denote a rotation of the whole cube (3 layers, if you want). The usual modificators can be used: for example, `y2` and `[u2]` both denote a 180 degrees rotation of the cube along the U/D axis.

- Inner layer moves: `M` = `R L' x'`, `E` = `U D' y'` and `S` = `F' B z`.

		
`M`	`E`	`S`

They cannot be used in official FMC solutions.

Appendix C

Last Layer Algorithms

Here is a list of last layer algorithms requiring 10 moves or less, modulo inverses and rotations. I have also excluded all the edge and corner 3-cycles, that are better learnt in that context.

6 to 9 Moves

ZBLLs (all edges oriented)
`R U R' U R U2 R'` (7)
`R U2 R2 U' R2 U' R2 U2 R` (9)
`R2 D L' B2 L D' R' U2 R'` (9)
`R B2 L2 D L D' L B2 R'` (9)
`F2 R2 L2 B2 D B2 R2 L2 F2` (9)

Other Algorithms
`F R U R' U' F'` (6)
`L F R' F R F2 L'` (7)
`R U R2 F R F2 U F` (8)
`R' U' R' F R F' U R` (8)
`R U R' U' R' F R F'` (8)
`R U2 R' U2 R' F R F'` (8)
`R U R' F' L' U' L F` (8)
`R U2 R2 F R F' R U2 R'` (9)
`R' F R U R' U' F' U R` (9)
`R' F R2 B' R2 F' R2 B R'` (9)
`R' B U2 B' U2 B' R2 B R'` (9)
`R B' R B2 L' B L B2 R2` (9)

10 Moves

ZBLLs (all edges oriented)
R U' B L U L' B2 R B R2
B2 L U L' B2 R D' R D R2
R2 U R2 D' F2 L2 U' L2 D F2
L' B L' D2 R F' R' D2 L2 B'
F U R U2 R' U R U R' F'
R U2 R' B' U R U R' U' B
R U' R' U2 R L U' R' U L'
R U' L U L2 D' B2 D R' L
R U' L' B2 U' B2 U B2 R' L
R B2 R2 U L U' R2 L' B2 R'

Other Algorithms
F U R U2 R' U F' L' U L
F U R U2 R' U' R U R' F'
F U R U2 R' U' F' L' U L
F U R U2 R' F' U' L' U2 L
F R B' R B R' U R' U' F'
F R U' B U B' U R' U' F'
F R U R' U' R U R' U' F'
F U R U' F' L F R' F' L'
F U R U' B R' F' R B' R'
F U R' U2 R2 U R2 U R F'
F U R2 D R' U' R D' R2 F'
F U R' U' R F' U' R' U2 R
F U R' F R F' R U' R' F'
R U R' U' B' R' F R F' B
R U2 R' U2 B' R' F R F' B
R U R2 F' U' F U R2 U2 R'
R U R' F D B' R' B D' F'
R U R' F2 D' B L' B' D F2
R U R' F2 B D' L' D F2 B'
R U R' B' R B U' B' R' B
R U L B L' U' L B' R' L'
R U2 F' U2 F R2 B' R2 B R'
R U2 R F2 L F L' F2 R2 F'
R U2 R2 F R F L F L' F
R U2 R' F2 L' B L' B' L2 F2
R U' F2 D2 L B2 D L' D F2
R U' F' L' U L F R U' R2
R U' L' U R' U2 B' U B L
R F' U' L' U F R' F' L F
R B U B2 U' R' U R B R'
R B U B' R' U' R' F R F'
L F L' R U R' U' L F' L'
R B R' F R B2 R B R2 F'
R B' R B R2 U2 F R' F' R
R L2 D' B' D B L B' R' L
R2 F2 R' U B U' B' R F2 R2
R2 F2 L D' F' D F L' F2 R2