Three-Cycles

Overview

Three-cycles (or 3-cycles) are particular permutations of cube pieces where only 3 pieces are permuted. Three-cycles are permutations of even-parity. From group theory, it is known that any permutation of the group of even-parity permutations or *Alternating Group* can be expressed as the composition (product) of 3-cycles – formally, they are *generators* for the group. This means that *any* member of the Alternating Group can be generated by composing 3-cycles only, hence the importance of generating a *complete* database of short (and possibly optimal) algorithms of 3-cycles.

Short algorithms of 3-cycles of pieces can be found by running algorithm templates in the Excel/VBA version of Algorithm Finder to give *seeds*. A seed algorithm is defined as a *short* algorithm that can generate a set of many other algorithms, by rotation/reflection, inversion, cyclic-shifting, or conjugation. Seed algorithms are *irreducible*, meaning that they can't be obtained from each other by inversion, cyclic-shifting or symmetry considerations alone (see: http://www.mementoslangues.fr/CubeDesign/CubeTheory/SeedAlgorithms.pdf).

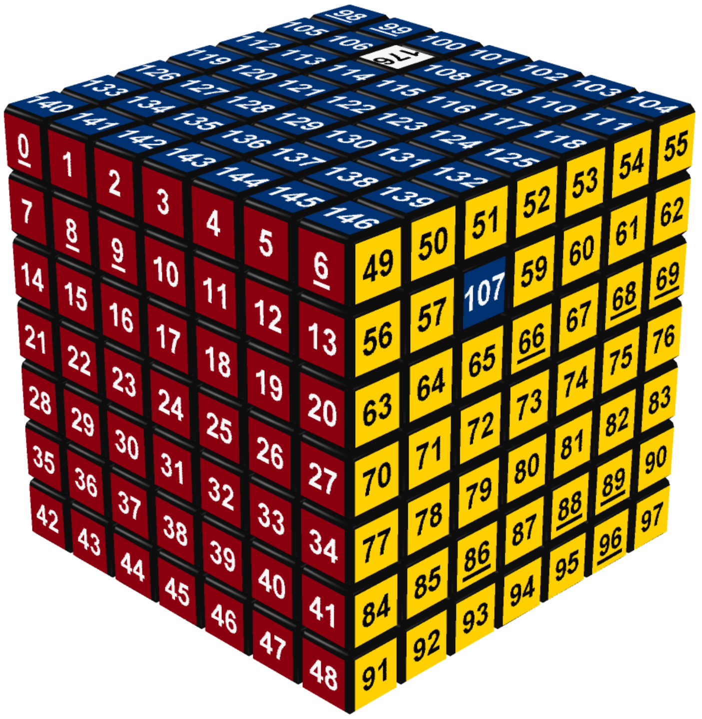
Seed algorithms are sorted out in stacks, where shortest algorithms are placed on top. Stacks of *scalable* seed algorithms have been built for 3-cycles of:

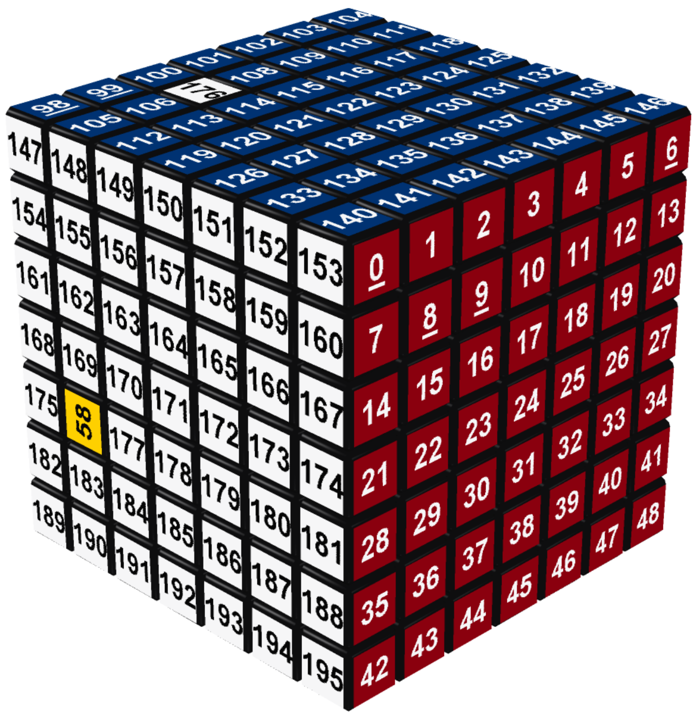
- Corners
- Midges
- Edges
- Corner-Centers
- Midge-Centers
- Edge-Centers

To generate a *complete* database of algorithms of 3-cycles, seed algorithms are picked up from the stack one at a time and transformed by rotation/reflection, inversion, cycle shifting and conjugation. Transformed algorithms are then further processed by selecting the shortest algorithm found before updating the database. This process is repeated until the database is complete.

Scalable Seed Algorithm Example – Edge-Centers 3-Cycle

3-Cycle of Edge-Centers Left (58 176 107)





NB' U' N3F' U NB U' N3F U = [NB', U' N3F' U]

Basic Group Theory

Group Theory – Useful Links
http://en.wikipedia.org/wiki/Permutation
http://en.wikipedia.org/wiki/Symmetric_group
http://en.wikipedia.org/wiki/Conjugacy_class
http://en.wikipedia.org/wiki/Commutator_subgroup
http://en.wikipedia.org/wiki/Coset
http://groupprops.subwiki.org/wiki/Cycle_type_of_a_permutation

By applying basic group theory to the cube, it can be inferred that (see links above for more details):

Conjugacy Class:

If A is an algorithm that generates a permutation of a given cycle type, then conjugate B A B' is also an algorithm that will generate another permutation of the *same* cycle type. This means that conjugating an algorithm will *not* change the cycle type of the associated permutation, or more explicitely if A generates a 7-cycle of corner-centers, then B A B' will also generate a 7-cycle of corner-centers. This is a very simple means of building a set of algorithms of a given cycle type from an already known 'seed' algorithm. The operation of conjugating an algorithm is also known as 'setup'. In most practical cases, setup moves are 1-, 2- or 3-move long. By using a set of different seed algorithms, it is possible to lower the number of needed setup moves from 3 to 2 or even 1.

Coset:
If A is an algorithm that generates a permutation of a given cycle type, then B A is also an algorithm that will generate another permutation, but which is generally of a *different* cycle type, although permutations of a same cycle type *may* be generated in *some* cases. This means that by inserting moves at the left, algorithms of a same cycle type *may* eventually be generated. Left added algorithms are usually 1-, 2- or 3-move long.

Symmetry:
If A is an algorithm that generates a permutation of a given cycle type, then transformed algorithm B(A), obtained by applying (anti)symmetry transformations on the set of letters <F, R, U, L, D, B> is also an algorithm that will generate another permutation of the *same* cycle type. This means that transforming an algorithm using all 48 possible cube (anti)symmetries will *not* change the cycle type of the associated permutations, or more explicitely if A generates a 5-cycle of edge-centers, then 48 more algorithms may be obtained by transforming letters.

Inversion:
If A is an algorithm that generates a permutation of a given cycle type, then inverted algorithm A' is also an algorithm that will generate another permutation of the *same* cycle type. This means that for each already known algorithm, one more can be added simply by inversion.

Cycle Shifting:
If A is an algorithm of length n that generates a permutation of a given cycle type, then the n algorithms obtained by cycle shifting A will also generate permutations of the *same* cycle type. Alg NR' U N3R U' NR U N3R' U' and shifted version U' NR' U N3R U' NR U N3R' will *both* give a 3-cycle of edge-centers, for example.

Using a combination of all these techniques, ie. conjugation + left add + symmetry + inversion + cycle shifting, it is usually possible to generate many N-cycles of a given type from just a *limited* number of seed algorithms. The problem now is how to find *short* and *irreducible* seeds. This is where algorithm templates may come into play…

Seeds Stack

Seed algorithms are placed in a stack (array) with shortest on top. The stack is partitioned into blocks of equal length algorithms. These are delimited by indices which serve as delimiters. A complete database of algorithms would then be generated from a stack of seeds by looking up seeds in a given block and processing them using cube (anti)symmetry properties, inversion, cycle-shifting and conjugation to give many more algorithms. Blocks with shorter seeds are processed first. Partitioning a seeds stack into blocks and searching in individual blocks is an effective way of lowering computing time.

Seeds can be conjugated with 1 move setup algorithms in a breadth-first search to generate more algorithms. For example, template X (Seed8) X' gives algorithms that can be 9 or 10 moves long and will add to the number of algorithms already present in database.

Seeds Stack – 3-Cycles	
Moves	Delimiters
8 moves (seed8)	← 0
	…
	← 1
10 moves (seed10)	← 2
	…
	← 3
12 moves (seed12)	← 4
	…
	← 5

DataBase – 3-Cycles	
Moves	Algorithms
8 moves	**(seed8)**
	–
9 moves	X (seed8) X'
	X (seed10) X'
10 moves	**(seed10)**
	X (seed10) X'
11 moves	X (seed10) X'
	X (seed12) X'
12 moves	**(seed12)**
	X (seed10) X'

Seed Algorithms Processing

All stickers of a given orbit of pieces are set to -1 on the mask (goal state). To search for algorithms, templates are first executed and end cube states compared to the goal state. If there is a match between states, then an algorithm has been found. The order of the permutation and the number of permuted pieces are computed from the end cube state. Algorithms of 3-cycles are then screened by setting *both* the order of the permutation and the number of permuted pieces to 3. These screened algorithms are added to a stack until all templates have been executed. At the end of the search, algorithms are further processed in 3 steps to give *irreducible* seeds by deleting 'equivalent' algorithms of the same length:

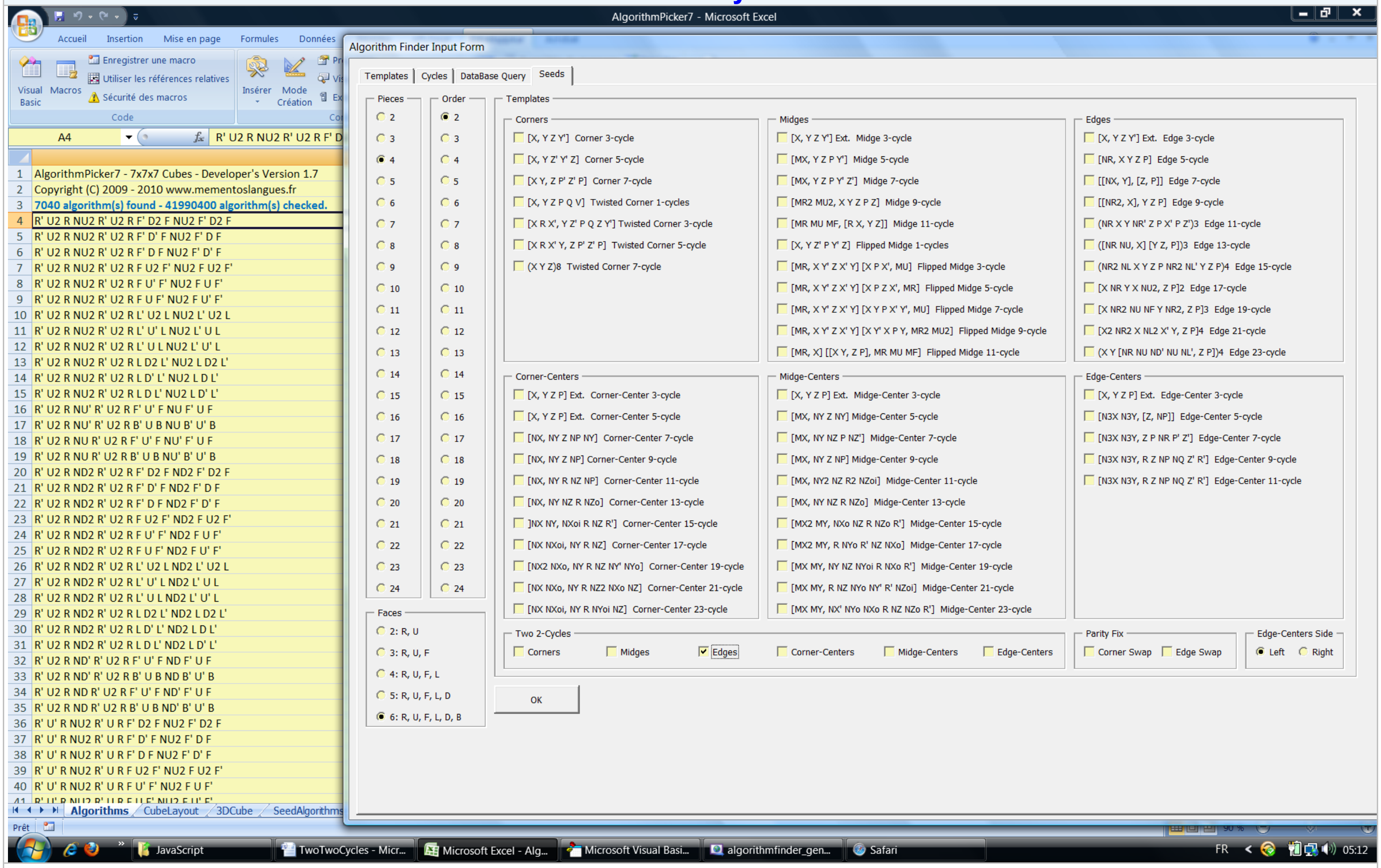
1. duplicates and their inverses
2. (anti)symmetry-duplicates and their inverses
3. shift-duplicates and their inverses

The screening process can be further refined by deleting 'trivial cases', ie. algorithms that differ from a given algorithm only by a permutation of basic moves on *opposed* faces in expressions like 'F B' or 'B F'.

Seeds are finally sorted out by number of moves to give blocks of equal length algorithms and by QTM-number of moves *inside* each block. The end result is that the first algorithms just on top of each block are the shortest in terms of QTM-moves for the block. This ensures that QTM-shortest algorithms of each block will be accessed first. The block with the shortest algorithms of all the stack is placed on top of the stack and will be visited first when generating the database.

Search for irreducible seed algorithms using Algorithm Finder – 7x7x7 Cube

Corners 3-cycles



7040 algorithms are further processed to give 777 seed algorithms

Algorithm Templates

Algorithms templates are the same for *all* pieces. Notice that only 3 of them are selected for Midges 3-cycle because in this particular case, all algorithms are 8, 9 or 10 moves long only.

Algorithm Templates – Corners 3-Cycles	
Template	Moves
[X, Y Z P]	8
[X, Y Z P Q]	10
[X Y, Z P Q]	10
[X, Y Z P Q V]	12
[X Y, Z P Q V]	12
Set of Moves – 18 Moves in Set	
<R, R', R2, U, U', U2, F, F', F2, L, L', L2, D, D', D2, B, B', B2>	
Number of Permuted Pieces / Stickers	Permutation Order
3 / 9	3

Algorithm Templates – Midges 3-Cycles	
Template	Moves
[X, Y Z Y']	8
[X, Y Z Y' Z']	10
[X Y, Z P Z']	10
Set of Moves – 36 Moves in Set	
<R, R', R2, U, U', U2, F, F', F2, L, L', L2, D, D', D2, B, B', B2,	
MR, MR', MR2, MU, MU', MU2, MF, MF', MF2, ML, ML', ML2, MD, MD', MD2, MB, MB', MB2>	
Number of Permuted Pieces / Stickers	Permutation Order
3 / 6	3

Algorithm Templates – Edges 3-Cycles	
Template	Moves
[X, Y Z Y']	8
[X, Y Z Y Z']	10
[X Y, Z P Z']	10
[X, Y Z P Y' Z']	12
[X, Y Z P Z' Y']	12
[X Y, Z P Z' P']	12
Set of Moves – 36 Moves in Set	
<R, R', R2, U, U', U2, F, F', F2, L, L', L2, D, D', D2, B, B', B2,	
NR, NR', NR2, NU, NU', NU2, NF, NF', NF2, NL, NL', NL2, ND, ND', ND2, NB, NB', NB2>	
Number of Permuted Pieces / Stickers	Permutation Order
3 / 6	3

Algorithm Templates – Corner-Centers 3-Cycles	
Template	Moves
[X, Y Z Y']	8
[X, Y Z Y Z']	10
[X Y, Z P Z']	10
[X, Y Z P Y' Z']	12
[X, Y Z P Z' Y']	12
[X Y, Z P Z' P']	12
Set of Moves – 36 Moves in Set	
<R, R', R2, U, U', U2, F, F', F2, L, L', L2, D, D', D2, B, B', B2,	
NR, NR', NR2, NU, NU', NU2, NF, NF', NF2, NL, NL', NL2, ND, ND', ND2, NB, NB', NB2>	
Number of Permuted Pieces / Stickers	Permutation Order
3 / 3	3

Algorithm Templates – Midge-Centers 3-Cycles	
Template	Moves
[X, Y Z Y']	8
[X, Y Z Y Z']	10
[X Y, Z P Z']	10
[X, Y Z P Y' Z']	12
[X, Y Z P Z' Y']	12
[X Y, Z P Z' P']	12
Set of Moves – 48 Moves in Set	
<R, R', R2, U, U', U2, F, F', F2, L, L', L2, D, D', D2, B, B', B2,	
MR, MR', MR2, MU, MU', MU2, MF, MF', MF2, ML, ML', ML2, MD, MD', MD2, MB, MB', MB2,	
NR, NR', NR2, NU, NU', NU2, NF, NF', NF2, NL, NL', NL2, ND, ND', ND2, NB, NB', NB2>	
Number of Permuted Pieces / Stickers	Permutation Order
3 / 3	3

Algorithm Templates – Edge-Centers 3-Cycles	
Template	Moves
[X, Y Z Y']	8
[X, Y Z Y Z']	10
[X Y, Z P Z']	10
[X, Y Z P Y' Z']	12
[X, Y Z P Z' Y']	12
[X Y, Z P Z' P']	12
Set of Moves – 48 Moves in Set	
<R, R', R2, U, U', U2, F, F', F2, L, L', L2, D, D', D2, B, B', B2,	
NR, NR', NR2, NU, NU', NU2, NF, NF', NF2, NL, NL', NL2, ND, ND', ND2, NB, NB', NB2,	
N3R, N3R', N3R2, N3U, N3U', N3U2, N3F, N3F', N3F2, N3L, N3L', N3L2, N3D, N3D', N3D2, N3B, N3B', N3B2>	
Number of Permuted Pieces / Stickers	Permutation Order
3 / 3	3

Seeds Stack Example – Corners 3-Cycle

Here below is a list of the first seeds in a stack of 3-cycles of corners, written in JavaScript code:

```
//JavaScript Code
//Three Consecutive Blocks in Stack: {8 movers}, {10 movers} and {12 movers}
//Stack of 336 Corner Seeds: AF.seedsArray_Corners[3]
AF.seedsArray_Corners[3] = new Array(
"Ui Li U Ri Ui L U R",
"U Li Ui Ri U L Ui R",
"Ui L2 U Ri Ui L2 U R",
"U L2 Ui Ri U L2 Ui R",
"Ui Li U R2 Ui L U R2",
"U Li Ui R2 U L Ui R2",
"Ui L2 U R2 Ui L2 U R2",
"U L2 Ui R2 U L2 Ui R2",
"R Ui F U Fi Ri F Ui Fi U",
"R U Bi Ui B Ri Bi U B Ui",
"R Li Ui Li U Ri Ui L U L",
"R Li Ui L U Ri Ui Li U L",
"R Li U Li Ui Ri U L Ui L",
"R Li U L Ui Ri U Li Ui L",
"R L Ui Li U Ri Ui L U Li",
"R L Ui L U Ri Ui Li U Li",
"R L U Li Ui Ri U L Ui Li",
"R L U L Ui Ri U Li Ui Li",
"Li Ui Li U R Ui L U Ri L",
"Li Ui L U R Ui Li U Ri L",
"Li U Li Ui R U L Ui Ri L",
"Li U L Ui R U Li Ui Ri L",
"L Ui Li U R Ui L U Ri Li",
"L Ui L U R Ui Li U Ri Li",
"L U Li Ui R U L Ui Ri Li",
"L U L Ui R U Li Ui Ri Li",
"Li Ui Li U Ri Ui L U R L",
"Li Ui L U Ri Ui Li U R L",
"Li U Li Ui Ri U L Ui R L",
"Li U L Ui Ri U Li Ui R L",
"L Ui Li U Ri Ui L U R Li",
"L Ui L U Ri Ui Li U R Li",
"L U Li Ui Ri U L Ui R Li",
"L U L Ui Ri U Li Ui R Li",
"R Ui Li U L2 Ri L2 Ui L U",
"R Ui L U L2 Ri L2 Ui Li U",
"R U Li Ui L2 Ri L2 U L Ui",
"R U L Ui L2 Ri L2 U Li Ui",
"R Li Ui L2 U Ri Ui L2 U L",
"R Li U L2 Ui Ri U L2 Ui L",
"R L Ui L2 U Ri Ui L2 U Li",
"R L U L2 Ui Ri U L2 Ui Li",
"R2 Ui F U Fi R2 F Ui Fi U",
"R2 Ui Li U Li R2 L Ui L U",
"R2 Ui Li U L R2 Li Ui L U",
"R2 Ui L U Li R2 L Ui Li U",
"R2 Ui L U L R2 Li Ui Li U",
"R2 U Li Ui Li R2 L U L Ui",
"R2 U Li Ui L R2 Li U L Ui",
"R2 U L Ui Li R2 L U Li Ui",
"R2 U L Ui L R2 Li U Li Ui",
```

DataBase Example – Corners 3-Cycle

Here below is a list of the first generated algorithms of a Database of 3-cycles of corners, as displayed on a web page when running the generator part of Algorithm Finder 7:

Algorithm Finder 7 --- 7x7x7 Cubes --- JavaScript Version1.8 --- Copyright (c) 2009 - 2010 mementoslangues
Generation of Algorithm Database
DataBase generated on Sat Apr 17 2010 15:23:28 GMT+0200 (Paris, Madrid (heure d'été))
Database of Corners 3-Cycles
DataBase is complete.
Elapsed Time (Hours:Minutes:Seconds) = 00:11:22

Seeds Stack Statistics:
Total Number of Seeds in Stack: 336
Number of Seeds of Length 8: 8 --- Percentage of Total Seeds in Stack: 2 %
Number of Seeds of Length 10: 70 --- Percentage of Total Seeds in Stack: 21 %
Number of Seeds of Length 12: 258 --- Percentage of Total Seeds in Stack: 77 %
Number of Blocks of Seeds in Stack: 3

DataBase Statistics:
Total Number of Indexed Algorithms in DataBase: 9072
Average Number of Moves per Algorithm: 8.7
Number of Algorithms of Length 8: 4752 --- Percentage of Total DataBase Algorithms: 52 %
Number of Algorithms of Length 9: 3024 --- Percentage of Total DataBase Algorithms: 33 %
Number of Algorithms of Length 10: 720 --- Percentage of Total DataBase Algorithms: 8 %
Number of Algorithms of Length 11: 432 --- Percentage of Total DataBase Algorithms: 5 %
Number of Algorithms of Length 12: 144 --- Percentage of Total DataBase Algorithms: 2 %

DataBase of Indexed Algorithms:
[78] R' D R' U2 R D' R' U2 R2 (9 moves) (0 -> 6 -> 48 -> 0)
[79] U2 B U' F2 U B' U' F2 U' (9 moves) (0 -> 6 -> 91 -> 0)
[80] U L U' R U L' U' R' (8 moves) (0 -> 6 -> 202 -> 0)
[81] U' R U' L2 U R' U' L2 U2 (9 moves) (0 -> 6 -> 42 -> 0)
[82] L' U' R' U L U' R U (8 moves) (0 -> 6 -> 196 -> 0)
[83] U' F2 U' B' U F2 U' B U2 (9 moves) (0 -> 6 -> 195 -> 0)
[84] B' R2 B' L2 B R2 B' L2 B2 (9 moves) (0 -> 6 -> 245 -> 0)
[85] F R B R' F' R B' R' (8 moves) (0 -> 6 -> 55 -> 0)
[86] F U' B' U F' U' B U (8 moves) (0 -> 6 -> 104 -> 0)
[87] B2 R2 B' L2 B R2 B' L2 B' (9 moves) (0 -> 6 -> 251 -> 0)
[88] L U' R' U L' U' R U (8 moves) (0 -> 6 -> 98 -> 0)
[89] L' B' L F' L' B L F (8 moves) (0 -> 6 -> 147 -> 0)
[90] L2 U' R' U L2 U' R U (8 moves) (0 -> 6 -> 293 -> 0)
[91] F U' B2 U F' U' B2 U (8 moves) (0 -> 6 -> 189 -> 0)
[92] U B' U' F' U B U' F (8 moves) (0 -> 6 -> 238 -> 0)
[93] U L U' R2 U L' U' R2 (8 moves) (0 -> 6 -> 287 -> 0)
[94] F U' B U F' U' B' U (8 moves) (0 -> 6 -> 244 -> 0)
[95] U B2 U' F' U B2 U' F (8 moves) (0 -> 6 -> 97 -> 0)
[102] L2 D2 L U2 L' D2 L U2 L (9 moves) (0 -> 146 -> 48 -> 0)
[103] R' F2 R' B' R F2 R' B R2 (9 moves) (0 -> 146 -> 91 -> 0)
[104] L D' F2 D L' D' L F2 L' D (10 moves) (0 -> 146 -> 202 -> 0)
[105] L2 B L' F2 L B' L' F2 L' (9 moves) (0 -> 146 -> 42 -> 0)
[106] R U2 R D2 R' U2 R D2 R2 (9 moves) (0 -> 146 -> 196 -> 0)
[107] D R' F2 R D' R' D F2 D' R (10 moves) (0 -> 146 -> 195 -> 0)
[108] L2 B2 L' F2 L B2 L' F2 L' (9 moves) (0 -> 146 -> 245 -> 0)
[109] L' B U2 B' L B L' U2 L B' (10 moves) (0 -> 146 -> 55 -> 0)
[110] R U2 R D R' U2 R D' R2 (9 moves) (0 -> 146 -> 104 -> 0)
[111] B' R U2 R' B R B' U2 B R' (10 moves) (0 -> 146 -> 251 -> 0)
[112] R' F2 R' B2 R F2 R' B2 R2 (9 moves) (0 -> 146 -> 98 -> 0)
[113] L2 D' L U2 L' D L U2 L (9 moves) (0 -> 146 -> 147 -> 0)
[114] B2 R U2 R' B R B' U2 B R' B (11 moves) (0 -> 146 -> 293 -> 0)
