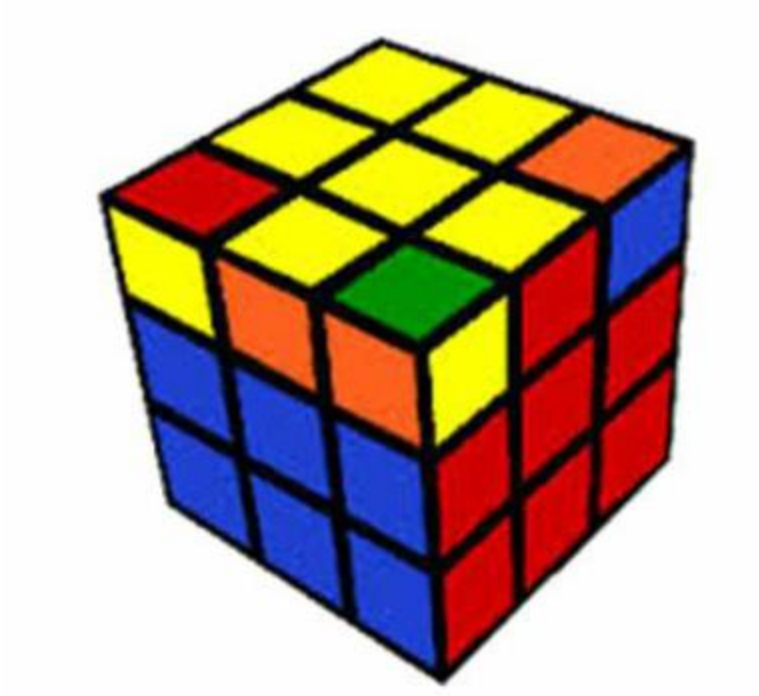
PLL ALGORITHMS

FOR PERSONAL USE ONLY



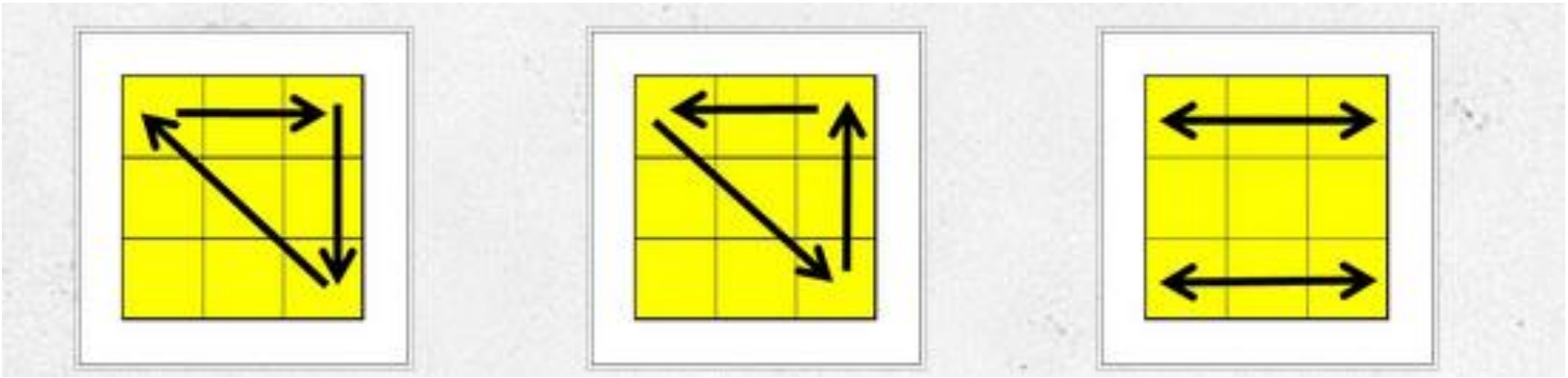
CONTENTS:

FS CUBING

Information

PLL is the fourth and final stage of the CFOP Method. PLL stands for Permuting Last Layer. This is probably one of the easiest steps of the method, because there are only 21 different algorithms. Although some cases may be hard to recognise at first, it just comes with practice and you should be able to learn full PLL in a couple of weeks at the most. I highly recommend learning full OLL and F2L first, so you have a starting point for the CFOP Method. I also recommend that you have learnt them all completely and can finger-trick them well. This is because PLL is the last step and in my opinion, the easiest step to learn, and once you have done OLL, you can go straight into PLL without much hesitation. For any algorithms displayed with something like r2, R2, do an M slice move like M2. This is much easier than having to do the complete algorithm. Remember: M is homologous to L, E is homologous to D and S is homologous to F. M Slices come with practice, as well as S and E slices, but S and E slices are very rarely ever used in cubing. I would use my less- dominant hand to do any M slice moves, and my dominant hand to do U, R or F moves in between M slices. As with OLL, I would learn 2-Look PLL first (link at the bottom of the page) to already give you 4 of the 21 algorithms.

Algorithms- Corner Permutations

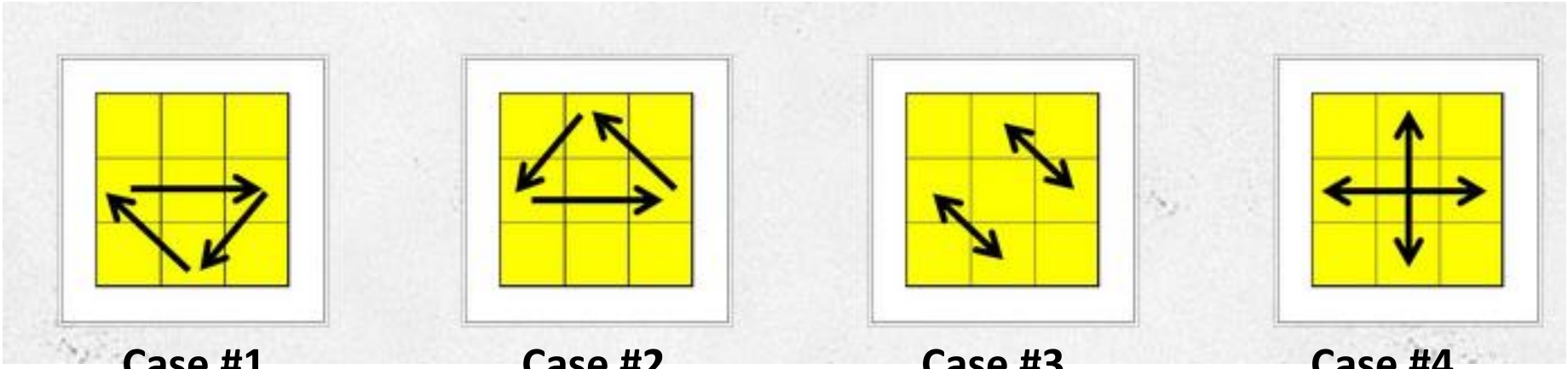


Case #1 **Case #2** **Case #3**

Case #1- (x), R', U, R', D2, R, U', R', D2, l2, (x)
Case #2- (x), R2, D2, R, U, R', D2, R, U', l
Case #3- l', U', L', U, R, U', L, U, R', U', L, U, R, U', L', U, (x')

FS CUBING

Algorithms- Edge Permutations



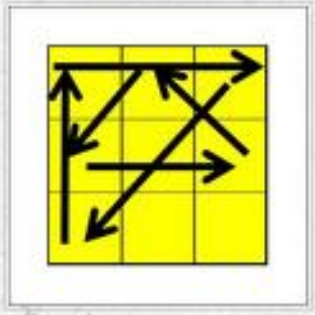
Case #1 **Case #2** **Case #3** **Case #4**

Case #1- R2', U, R, U, R', U', R', U', R', U, R'
Case #2- R2, U', R', U', R, U, R, U, R, U', R
Case #3- R', U', R, U', R, U, R, U', R', U, R, U, R2, U', R', U2
Case #4- r2, R2', U, r2, R2, U2, r2, R2, U, r2, R2

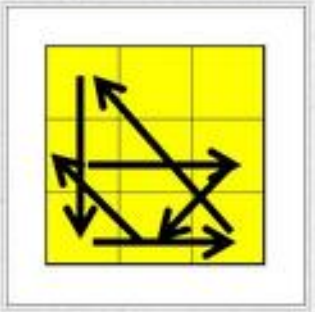
(For Case #4, you could also do: M2, U, M2, U2, M2, U, M2)

FS CUBING

Algorithms- G Permutations

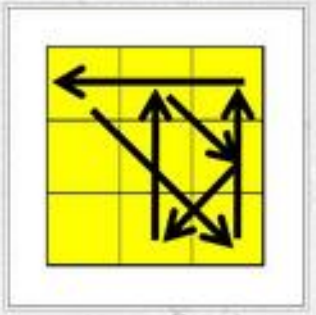


Case #1



Case #2

Case #3

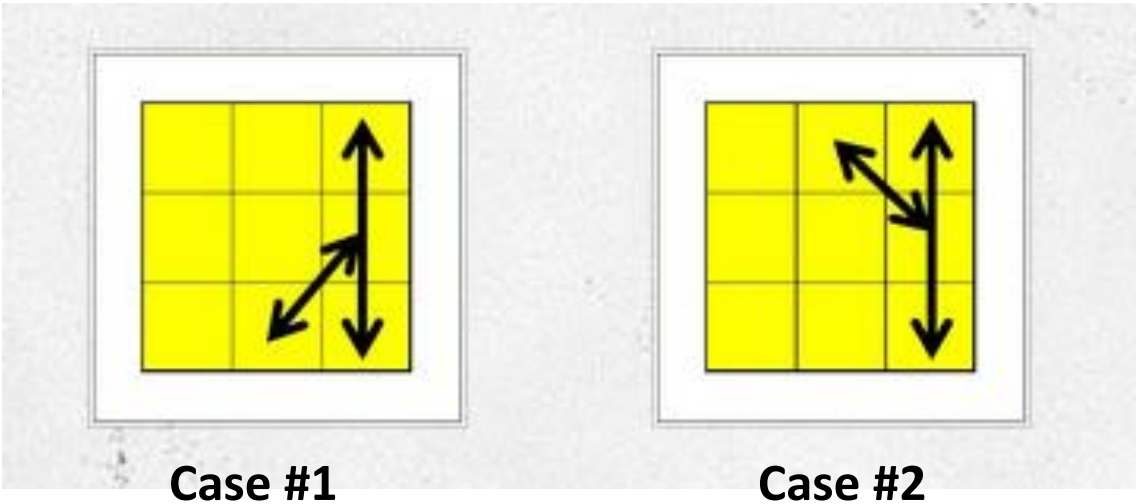


Case #4

Case #1- R2', u, R', U, R', U', R, u', R2, (y'), R', U, R
Case #2- R2, u', R, U', R, U, R', u, R2', (y), R, U', R'
Case #3- R, U, R', (y'), R2, u', R, U', R', U, R', u, R2
Case #4- L', U', L, (y'), R2', u, R', U, R, U', R, u', R2

FS CUBING

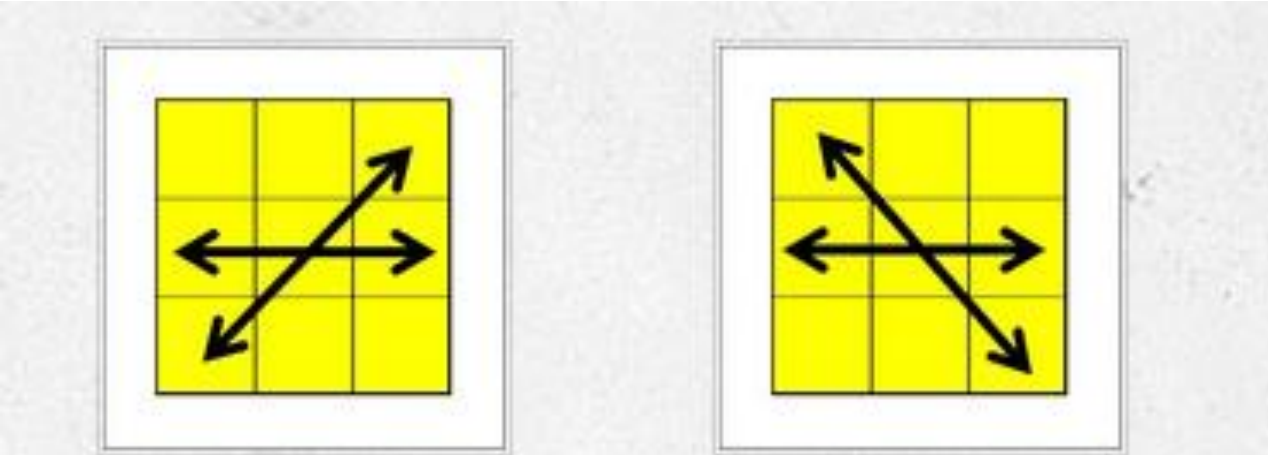
Algorithms- J Permutations



Case #1 **Case #2**

Case #1- R, U, R', F', R, U, R', U', R', F, R2, U', R', U'
Case #2- F2', L', U', r, U2', l', U, R', U', R2, (x2)

Algorithms- N Permutations



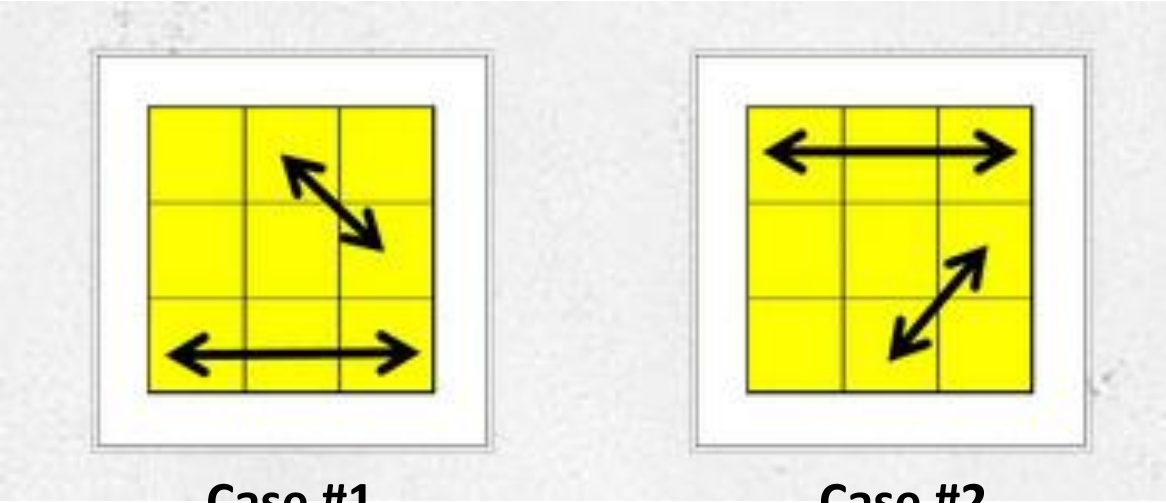
Case #1

Case #2

Case #1- R, U', L, U2', R', U, L', R, U', L, U2', R', U, L'
Case #2- L', U, R', U2', L, U', L', R, U, R', U2', L, U', R

FS CUBING

Algorithms- R Permutations



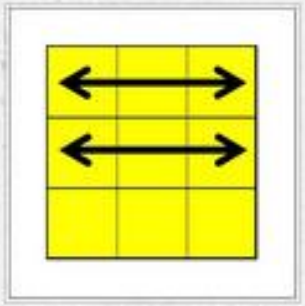
Case #1

Case #2

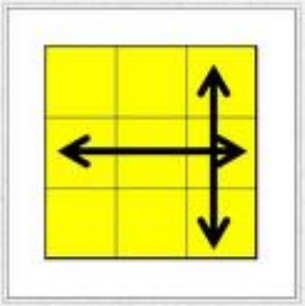
Case #1- R, U2', R', U2, R, B', R', U', R, U, l, U, R2', F, (x)
Case #2- R', U2, R, U2', R', F, R, U, R', U', R', F', R2, U'

FS CUBING

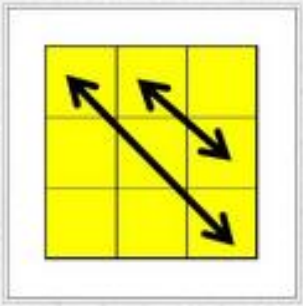
Algorithms- Other Permutations



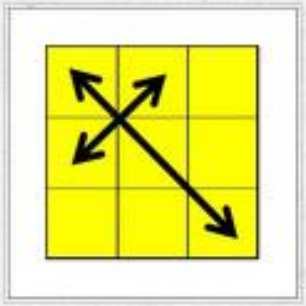
Case #1



Case #2



Case #3



Case #4

Case #1- R', U, R, U', R2, F', U', F, U, (x), R, U, R', U', R2, B', (x')
Case #2- R, U, R', U', R', F, R2, U', R', U', R, U, R', F'
Case #3- R', U, R', U', (x2, y), R', U, R', U', l, R, U', R', U, R, U, (x)
Case #4- F, R, U', R', U', R, U, R', F', R, U, R', U', l', U, R, U', (x')