

## First Two Layers (F2L)

After the cross, the next step is F2L (First Two Layers), arguably the most important step of Fridrich Method. This step completes the first two layers by fixing the four corner-edge pairs (slots) between the cross edges in four steps, one slot at a time. There are 41 basic "algorithms" to insert a corner and its corresponding edge into their correct spot in the correct orientation. However, most of these are relatively short and intuitive compared to the last layer algorithms (hence the quotation marks).

Because the four pairs can be solved in any order, F2L is a relatively flexive step and allows for many techniques that reduce the number of moves. Rather than blindly applying the algorithms, it is important to understand how each one works and to be able to apply them from all directions. If you are new to F2L, consider learning it intuitively first from [Doug Reed's guide](#) and then comparing your algorithms with my list or with those at [cubeloop.com](#).

This page first lists algorithms for each of the 41 standard F2L cases with target slot FR. Each one is written as I often perform it, making ample use of double layer and whole cube turns (see [Notation](#)). Although multiple algorithms are often provided for one case, it is only necessary to know one. The rest of the page explains some techniques that can be used without learning any extra algorithms. To really master F2L, I recommend that you read and work on these techniques as you are learning the algorithms.

More advanced techniques, requiring new algorithms, are covered in [Advanced F2L](#).

## 41 Standard F2L Cases

### 4 basic patterns

Almost all basic F2L algorithms can be broken down into two stages: 1) place the corner-edge pair in one of the following four basic patterns; 2) solve the basic pattern. It is therefore essential that you first learn to recognize the following four cases.

Code	Pattern	Algorithm	Code	Pattern	Algorithm
I1		D'wL'UL	I2		URU'R'
T1		RUR'	T2		y'R'U'R'/yL'U'L

Whenever an algorithm follows this two-step pattern, the only part we need to memorize is the first step, which usually takes only 3 or 4 moves. I have italicized the first step when applicable. Of course, we eventually need to know the second step as well to gain speed, but this easily comes with practice.

Corner and edge on U, corner points to side (except K)

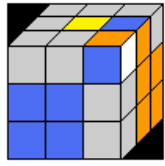
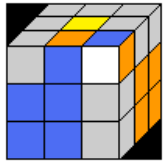
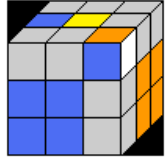
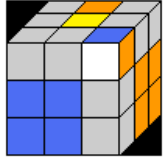
For the purpose of this page, a first-layer corner "points" in the direction of its first-layer sticker. Since I use blue as the cross colour, a corner pointing to the side would mean, for me, that the blue corner sticker is on one of the side faces.

Most of these cases have the two-step pattern described above and follow the following simple rules:

- Define the "target edge" to be the position relative to the corner where the edge can be placed, without being flipped, to create one of the basic patterns. This means that, if the corner and the edge have the same colour on top, we reduce to I1 or I2. If they are different, we reduce to T1 or T2.
- The first two moves are always either U'R or UF' (using double layer, DwR'), both of which brings the corner to the bottom two layers. In most cases, one or the other leaves both the edge and the target edge on U. Apply this one.
- Now turn the U layer until the edge is where the target edge was. When you bring back the corner to U, the corner-edge pair should be in a basic pattern.

For example, in Q1, the corner and the edge have the same colour on top, so we reduce to I, which means that the target edge is UF in the position shown. DwR' is therefore the correct opening, U' moves the edge to the target, and R brings back the corner, completing the first step. These solutions should all feel obvious after some practice.

Code	Pattern	Algorithm	Code	Pattern	Algorithm
Q1		$DwR'U'R-D2'wL'UL$	Q2		$U'RUR'-U2RU'R'$
S1		$DwR'U2R-D2'wL'UL$	S2		$U'RU2R'-U2RU'R'$
J1		$DwR'U2R-D'wRUR'$	J2		$U'RU2R'-DwR'U'R$

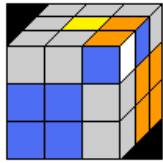
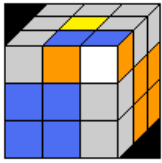
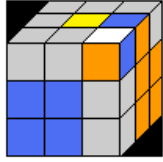
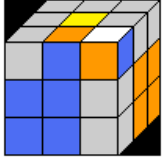
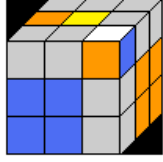
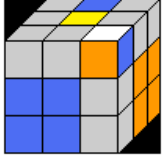
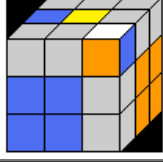
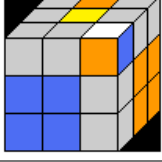
L1		$U'RUR'R'-URUR'$	L2		$DwR'UR-U'R'U'R$ $yUL'UL-U'L'U'L$
R1		$U'RUR'R'-URUR'$	R2		$U'RUR'R'-DwR'U'R$ $URUR'-UDwR'U'R$

Note that, for R1 and R2, both  $U'R$  and  $DwR'$  keep both the edge and the target edge on U. We choose the opening that allows for more finger tricks or has less cube rotations.

For K1 and K2, neither opening keeps both the edge and the target edge on U. We group this together with the second group.

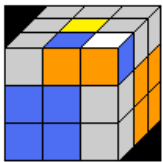
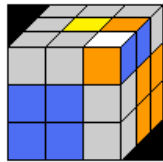
Corner and edge on U, corner points up (and K)

We now hold the edge in the first two layers and keep the corner on U. For U and V, the edge is aligned with the first two layers, hidden in the first two layers, and the pair is reduced to I. The alternate algorithms provided for V1 and V2, which are sometimes useful from different angles, do not follow this pattern.

Code	Pattern	Algorithm	Code	Pattern	Algorithm
K1		$RU'R'-U'D'wL'U'L$	K2		$y'R'UR-U'D'wRUR'$
N2		$RU2'R'-URUR'$	N1		$y'R'U2'R-UR'UR'$
U1		$Dw'L'U2L-U'L'UL$	U2		$URU2'R'-URU'R'$
V1		$U2'RUR'-URU'R'$ $RU'R'-U2RUR'$	V2		$UDwR'U'R-U'R'UR$ $yL'UL-U2'L'U'L$ $U2'RUR'-y'R'UR'$

Note the similarity in the first steps of K1 and N2 and of K2 and N1.

The last case, M, can be reduced to I or T. However, the optimal solution (first line) does not follow the two-step pattern. This is one of the few F2L sequences with no intuitive explanation.

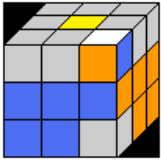
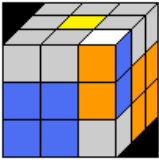
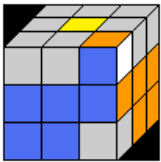
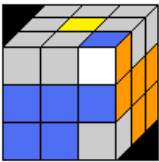
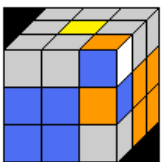
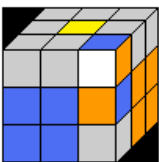
Code	Pattern	Algorithm	Code	Pattern	Algorithm
M1		$U2'R2U2'R'U'RU'R2'$ $RUR'U2RUR'U'RUR'$ $RU'R'U'RUR'U2RUR'$	M2		$yU2'L2U2LUL'UL2$ $U'R'FRF'-U'F'UF'$

Corner in place, Edge on U

Code	Pattern	Algorithm	Code	Pattern	Algorithm
A1		$URU'R'-Dw'L'UL$	A2		$Dw'L'UL-DwRU'R'$ $U'R'FRF'-RUR'$
E1		$yL'UL-U'L'UL$ $RU2'R'-F'U2F$	E2		$RU'R'-URU'R'$
F1		$U'RUR'-U'RUR'$	F2		$yUL'U'L-UL'U'L$ $UR'FRF'-URU'$

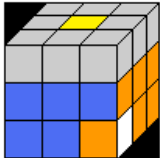
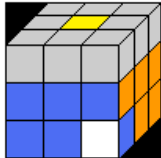
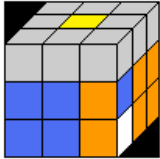
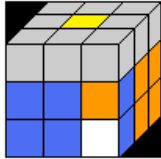
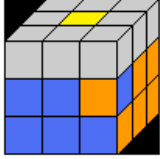
Corner on U, Edge in place

G and H differ in the first step only in the direction of the second quarter turn U. G is reduced to I, and H is reduced to T. B can be reduced to either and can be used to control the edge orientation of the last layer.

Code	Pattern	Algorithm	Code	Pattern	Algorithm
B1		$(URU'R') \times 3$ $(RUR'U') \times 3$ $y'(RB'R'B) \times 3$	B2		$U'R'FRF'-RU'R'$ $RU'R'-F'U2F$
G1		$URUR'-U2'RUR'$ $U'RU2'R'-URUR'$ $yUL'UL-U2'L'UL$	G2		$U'RU'R'-U2RU'R'$
H1		$yUL'U'L-U'y'RUR'$ $yzRU'R'U-R'FRF'$	H2		$U'RU'R'-DwR'U'R$

Corner and edge in place

The optimal sequences for all cases here are difficult to understand intuitively. C1 and C2 can be reduced to U2 and N2 by RUR' and RU'R', respectively. The first three moves of the first algorithm for A0 reduces the case to J2. The second can be used for last layer edge control.

Code	Pattern	Algorithm	Code	Pattern	Algorithm
C1		yL2U2LUL'ULU2L	C2		R2U2'R'U'RU' R'U2R'
D1		y'R'UR'Dw'L'ULF2	D2		RU'RUy'LU'L'B 2'
A0		RU2'R'URU2'R'DwR' U'R RUR'FUDwR2U'R2'D w'R'F'R			