

# Lars Vandenberg's CubeZone

Belgium's No. 1 speedcuber

[Algorithms](#) [ZB F2L](#) | [OLL](#) | [PLL](#) | [COLL](#) **Methods** [Square-1](#) **Tools** [ImageCube](#) | [ImageRevenge](#) **Media** [News articles](#) | [Videos](#) **Miscellaneous** [About me](#)  
[Overview](#) | [Notation](#) | [Step 1](#) | [Step 2](#) | [Step 3](#) | [Step 4](#) | [Step 5](#) | [Step 6](#)

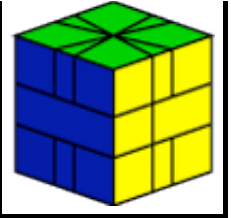
## Square-1 solution method - Overview

This section describes the method I use for solving Square-1. The various steps of the solution are listed below together with a short description. Throughout the solution I will use a certain notation for the move sequences, which is explained further on the [notation page](#).

Description	Result
<p><a href="#">Step 1: Make both layers square</a></p> <p>There are two ways of completing this step: I provide you with a simple edge-pairing technique that is suited for people who are just starting off. The more advanced can gradually learn the diagram of shapes which enables you to make both layers square in the optimal number of moves. With the advanced method, you can always complete this step in 7 twists, although you'd rarely need more than 5 twists.</p>	
<p><a href="#">Step 2: Bring the corners in their correct layer</a></p> <p>This is a short and intuitive step. The idea is to bring all corners which have the same color on top in the same layer. This doesn't require any algorithms and can always be done in 3 twists or less. I'll still provide you with some tips to make this step quicker.</p>	
<p><a href="#">Step 3: Bring the edges in their correct layer</a></p> <p>This is also a short step and it only requires 7 algorithms.</p>	
<p><a href="#">Step 4: Permute the corners within their layer</a></p> <p>Once again, this is also a short step that can be completed very quickly. It requires 5 easy algorithms to be able to permute the corners of both layers in one look.</p>	
<p><a href="#">Step 5: Permute the edges within their layer</a></p> <p>This is the hardest step to be able to do fast. The more algorithms you know, the better you get at it but there are plenty of sequences to learn and they are quite lengthy as well.</p>	

**Step 6: Correct middle layer and swap layers if necessary**

For this step you only need to know some simple sequences to flip the middle layer and to swap the layers.

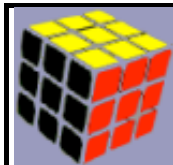


---

This page is maintained by [Lars Vandenberg](#)



Last update on 28th May 2007



# Lars Vandenberg's CubeZone

Belgium's No. 1 speedcuber

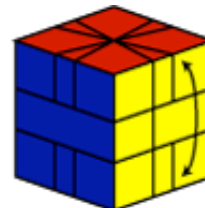
[Algorithms](#) [ZB F2L](#) | [OLL](#) | [PLL](#) | [COLL](#) **Methods** [Square-1](#) **Tools** [ImageCube](#) | [ImageRevenge](#) **Media** [News articles](#) | [Videos](#) **Miscellaneous** [About me](#)  
[Overview](#) | [Notation](#) | [Step 1](#) | [Step 2](#) | [Step 3](#) | [Step 4](#) | [Step 5](#) | [Step 6](#)

## Square-1 solution method - Notation

There are two types of moves you can do on Square-1:



*Turning the bottom and top layers*



*Twisting the right hand side 180°*

Each algorithm or move sequence on Square-1 is made up of turning the bottom and top layers, twisting the right hand side, turning the bottom and top layers, twisting the right hand side, and so on.

Since the small pieces in the bottom and top layers have a 30° angle and the large pieces have a 60° angle, every turn of the bottom and top layer will be a multiple of 30°. Hence the easiest way to write down a turn of the bottom and top layers is like a tuple:

**(a,b)** where **a** denotes the multiple of 30° you have to turn the top layer and **b** denotes the multiple of 30° you have to turn the bottom layer.

Positive values of **a** and **b** are used for clockwise turns, whereas negative values mean anti-clockwise turns.

A 180° twist of the right hand side is simply denoted by a forward slash:

/ means twisting the right hand side 180°.

So a complete algorithm may look like this:

/ (-3,0) / (3,3) / (0,-3) /

This algorithm swaps two adjacent corners in both layers when the puzzle is in the cube shape. The correct way to perform this algorithm would be to:

- / twist the right hand side 180°,
- (-3,0)** turn the top layer 90° anti-clockwise,
- / twist the right hand side 180°,
- (3,3)** turn both the bottom and top layers 90° clockwise,
- / twist the right hand side 180°,
- (0,-3)** turn the bottom layer 90° anti-clockwise
- / and finally twist the right hand side 180°.

If you're having trouble working out how far you have to turn the bottom and top layers, you can use the following rule of thumb: a **small piece** counts as **1** and a **large piece** counts as **2**. So if you have to perform **(2,0)**, it means that you either have to turn one large piece past the middle of the puzzle or two small pieces past the middle of the puzzle. Use the separation line between the two pieces of the middle layer as a guide.



# Lars Vandenberg's CubeZone

Belgium's No. 1 speedcuber

[Algorithms](#) [ZB F2L](#) | [OLL](#) | [PLL](#) | [COLL](#) **Methods** [Square-1](#) **Tools** [ImageCube](#) | [ImageRevenge](#) **Media** [News articles](#) | [Videos](#)

**Miscellaneous** [About me](#)

[Overview](#) | [Notation](#) | [Step 1](#) | [Step 2](#) | [Step 3](#) | [Step 4](#) | [Step 5](#) | [Step 6](#)

## Square-1 solution method - Step 1

### Make both layers square

One of the unique and interesting properties of the Square-1 is that it changes shape when you scramble it. Trying to solve the scrambled puzzle without making it into a cube first can prove to be a tough task since a lot of shapes have very limited options to move pieces around. The state where both layers are square is much more manoeuvrable and it allows us to recognise more easily where each piece belongs.

With this method, the long term goal is to be able to make both layers square in the optimal number of twists of the middle layer. But first we're going to investigate the various shapes that one layer can have, and work out which combinations of bottom layer shapes and top layer shapes are possible.

### Shapes that one layer can have

A layer can have various combinations of corners (large pieces) and edges (small pieces). There are a few restrictions however. Let's say **C** is the number of corners and **E** is the number of edges. Since the inner angle of all pieces must add up to 360°, we know that  $60C + 30E = 360$  or simpler:

$$2C + E = 12 \quad (\text{constraint 1})$$

There are also only 8 small pieces in total and only 8 large pieces in total:

$$0 \leq C \leq 8 \quad (\text{constraint 2})$$

$$0 \leq E \leq 8 \quad (\text{constraint 3})$$

If we know consider all possible values for **C** and calculate the value for **E** from it using constraint 1, we get the following results:

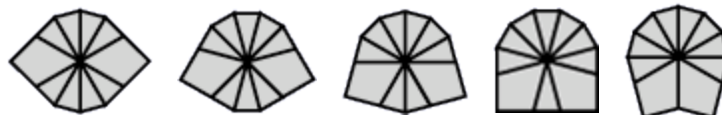
- C=0, E=12
- C=1, E=10
- C=2, E=8
- C=3, E=6
- C=4, E=4
- C=5, E=2
- C=6, E=0
- C=7, E=-2
- C=8, E=-4

If we discard all possibilities with an invalid value of **E** using constraint 3, only the following 5 options remain that satisfy all constraints:

- 2 corners and 8 edges
- 3 corners and 6 edges
- 4 corners and 4 edges
- 5 corners and 2 edges
- 6 corners and 0 edges

One can now work out all possible arrangements of corners and edges for each subcase, which leads to the following **29 shapes**:

**2 corners and 8 edges**  
(5 shapes)



4-4

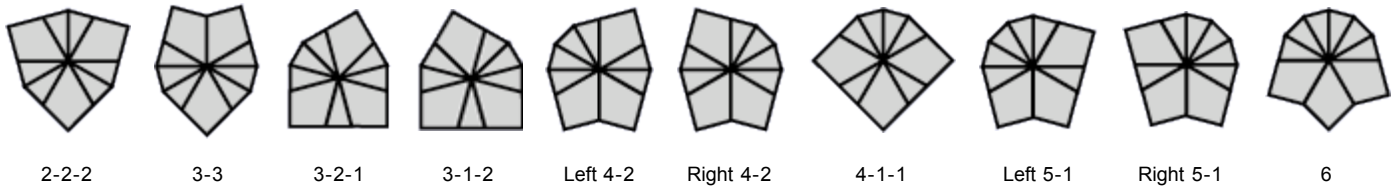
5-3

6-2

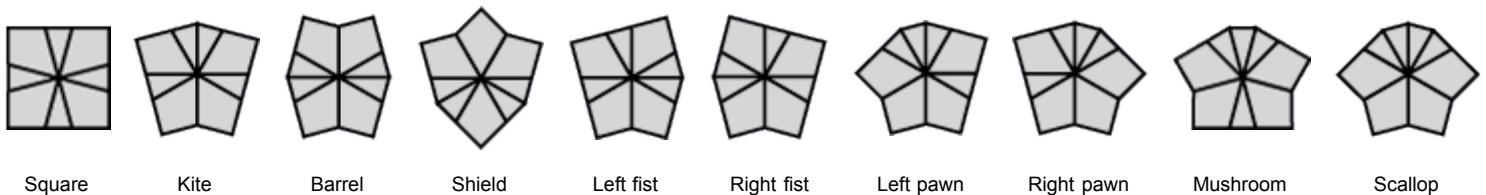
7-1

8

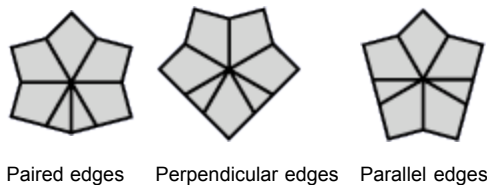
**3 corners and 6 edges**  
(10 shapes)



**4 corners and 4 edges**  
(10 shapes)



**5 corners and 2 edges**  
(3 shapes)



**6 corners and 0 edges**  
(1 shape)



**Combining shapes in both layers**

Now that we've worked out all possible shapes one layer can have, we can investigate all shapes the whole puzzle can have by considering all possible combinations of 2 shapes (one shape of the top layer and one shape of the bottom layer). Of course, we don't take the state of the middle layer into account. Since the total amount of corners and edges must at all times be exactly 8, we know that:

- if the top layer has **6 corners and 0 edges**, the bottom layer has to have **2 corners and 8 edges**.
- if the top layer has **5 corners and 2 edges**, the bottom layer has to have **3 corners and 6 edges**.
- if the top layer has **4 corners and 4 edges**, the bottom layer has to have **4 corners and 4 edges**.
- if the top layer has **3 corners and 6 edges**, the bottom layer has to have **5 corners and 2 edges**.
- if the top layer has **2 corners and 8 edges**, the bottom layer has to have **6 corners and 0 edges**.

The last two cases in the list contain the same shapes as the first two cases if you put the cube upside down. So from now on we will only talk about 3 possible configurations:

- **6 corners** in the top layer (1 case) and **2 corners** in the bottom layer (5 cases) which adds up to **5 cases** in total
- **5 corners** in the top layer (3 cases) and **3 corners** in the bottom layer (10 cases) which adds up to **30 cases** in total
- **4 corners** in the top layer (10 cases) and **4 corners** in the bottom layer (10 cases) which adds up to **55 cases** in total



# Lars Vandenberg's CubeZone

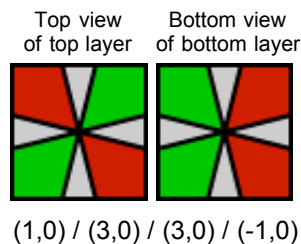
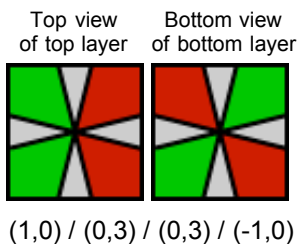
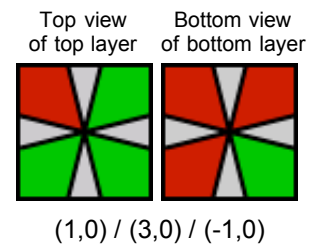
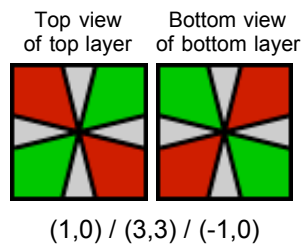
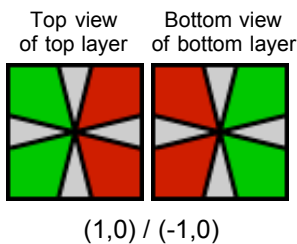
Belgium's No. 1 speedcuber

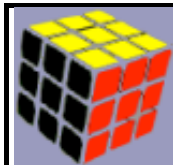
[Algorithms](#) [ZB F2L](#) | [OLL](#) | [PLL](#) | [COLL](#) **Methods** [Square-1](#) **Tools** [ImageCube](#) | [ImageRevenge](#) **Media** [News articles](#) | [Videos](#) **Miscellaneous** [About me](#)  
[Overview](#) | [Notation](#) | [Step 1](#) | **Step 2** | [Step 3](#) | [Step 4](#) | [Step 5](#) | [Step 6](#)

## Square-1 solution method - Step 2

### Bring the corners in their correct layer

This step is very similar to Geatan Guimond's method for solving the corners. We want to have all corners with the same top color in the same layer. The best way to do this is to turn the top layer by 30° degrees first in order to keep both layers square after each twist of the middle layer. If we now restrict our moves to quarter (90°) and half turns (180°) of either layer, the puzzle will more or less behave like a 2x2x2 Rubik's cube since each corner will "stick" to one of its adjacent edges. If you keep these things in mind, you should be able to solve this step easily. If you can't seem to work it out, or you think you're using too much moves, you can use the move sequences below to help you out.





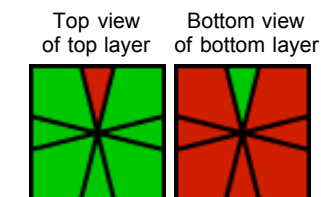
# Lars Vandenberg's CubeZone

Belgium's No. 1 speedcuber

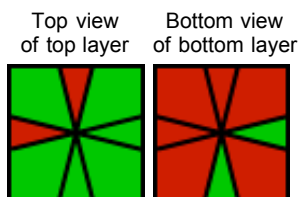
[Algorithms](#) [ZB F2L](#) | [OLL](#) | [PLL](#) | [COLL](#) **Methods** [Square-1](#) **Tools** [ImageCube](#) | [ImageRevenge](#) **Media** [News articles](#) | [Videos](#) **Miscellaneous** [About me](#)  
[Overview](#) | [Notation](#) | [Step 1](#) | [Step 2](#) | **Step 3** | [Step 4](#) | [Step 5](#) | [Step 6](#)

## Square-1 solution method - Step 3

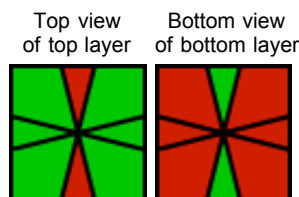
Bring the edges in their correct layer



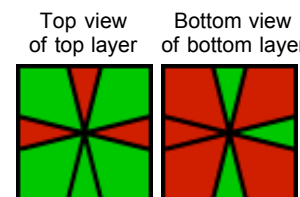
$(0,-1) / (-3,0) / (4,1) / (-4,-1) / (3,0) / (0,-1)$



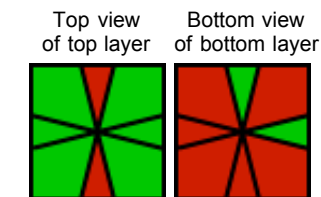
$(1,0) / (-3,0) / (-1,-1) / (4,1) / (-1,0)$



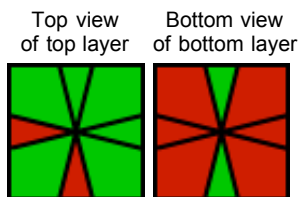
$(0,-1) / (1,1) / (-1,0)$



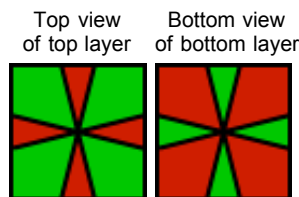
$(0,-1) / (1,4) / (-1,-4) / (-3,0) / (4,1) / (-1,0)$



$(0,-1) / (3,0) / (-3,0) / (1,-2) / (0,3) / (-1,0)$

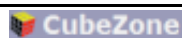


$(0,-1) / (3,0) / (0,-3) / (1,-2) / (3,0) / (-1,0)$



$(0,-1) / (1,1) / (3,3) / (-1,-1) / (0,1)$

This page is maintained by [Lars Vandenberg](#)



Last update on 20th November 2005



# Lars Vandenberg's CubeZone

Belgium's No. 1 speedcuber

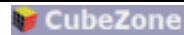
[Algorithms](#) [ZB F2L](#) | [OLL](#) | [PLL](#) | [COLL](#) **Methods** [Square-1](#) **Tools** [ImageCube](#) | [ImageRevenge](#) **Media** [News articles](#) | [Videos](#) **Miscellaneous** [About me](#)  
[Overview](#) | [Notation](#) | [Step 1](#) | [Step 2](#) | [Step 3](#) | **Step 4** | [Step 5](#) | [Step 6](#)

## Square-1 solution method - Step 4

### Permute the corners within their layer

<p>Top view of top layer</p>	<p>Bottom view of bottom layer</p>	<p>Top view of top layer</p>	<p>Bottom view of bottom layer</p>	<p>Top view of top layer</p>	<p>Bottom view of bottom layer</p>	<p>Top view of top layer</p>	<p>Bottom view of bottom layer</p>
$/ (3, -3) / (3, 0) / (-3, 0) / (0, 3) / (-3, 0) /$		$/ (3, 3) / (3, 0) / (3, 3) / (3, 0) / (3, 3) /$		$/ (3, -3) / (0, 3) / (-3, 0) / (3, 0) / (-3, 0) /$		$/ (3, 3) / (0, 3) / (3, 3) / (0, 3) / (3, 3) /$	
<p>Top view of top layer</p>	<p>Bottom view of bottom layer</p>	<p>Top view of top layer</p>	<p>Bottom view of bottom layer</p>	<p>Top view of top layer</p>	<p>Bottom view of bottom layer</p>	<p>Top view of top layer</p>	<p>Bottom view of bottom layer</p>
$/ (-3, 0) / (3, 3) / (0, -3) /$		$/ (0, 3) / (0, -3) / (0, 3) / (0, -3) /$		$/ (3, 0) / (-3, 0) / (3, 0) / (-3, 0) /$		$/ (3, -3) / (3, -3) /$	

This page is maintained by [Lars Vandenberg](#)



Last update on 20th November 2005





# Lars Vandenberg's CubeZone

Belgium's No. 1 speedcuber

[Algorithms](#) [ZB F2L](#) | [OLL](#) | [PLL](#) | [COLL](#) **Methods** [Square-1](#) **Tools** [ImageCube](#) | [ImageRevenge](#) **Media** [News articles](#) | [Videos](#) **Miscellaneous** [About me](#)  
[Overview](#) | [Notation](#) | [Step 1](#) | [Step 2](#) | [Step 3](#) | [Step 4](#) | **Step 5** | [Step 6](#)

## Square-1 solution method - Step 5

Below you can find algorithms for each case. First the case in cycle notation is listed, followed by the actual algorithm. The numbers between square brackets denote the number of twists and the the number of turns respectively. The algorithms are grouped according to the number of edge swaps that need to be done in each layer. You should keep in mind that a 2-cycle counts as 1 edge swap, a 3-cycle counts as 2 edge swaps and a 4-cycle is the equivalent of 3 edge swaps.

### 1 edge swap in U layer, 0 edge swaps in D layer

Top view of top layer      Bottom view of bottom layer



(UR UF)  
 / (-3,0) / (0,3) / (0,-3) / (0,3) / (2,0) / (0,2) / (-2,0) / (4,0) / (0,-2) / (0,2) / (-1,4) / (0,-3) / (0,3)  
 [13|27]



(UR UL)  
 / (3,3) / (-1,0) / (2,-4) / (4,-2) / (0,-2) / (-4,2) / (1,-5) / (3,0) / (3,3) / (3,0)  
 [10|26]

### 2 edge swaps in U layer, 0 edge swaps in D layer

Top view of top layer      Bottom view of bottom layer



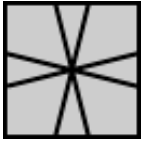
(UR UB) (UL UF)  
 / (3,3) / (0,3) / (1,1) / (-1,-4) / (-3,-3) /  
 [6|15]



(UR UL) (UF UB)  
 / (3,-3) / (3,-3) / (0,1) / (-3,3) / (-3,3) / (-1,0)  
 [6|16]






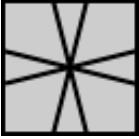

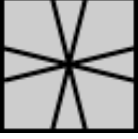
(UR UL UB)  
 / (3,0) / (1,0) / (0,-3) / (-1,0) / (-3,0) / (1,0) / (0,3) / (-1,0)  
 [8|16]




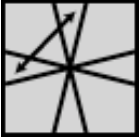
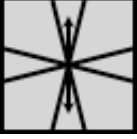
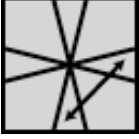


(UR UB UL)  
 (1,0) / (0,-3) / (-1,0) / (3,0) / (1,0) / (0,3) / (-1,0) / (-3,0) /  
 [8|16]

### 3 edge swaps in U layer, 0 edge swaps in D layer


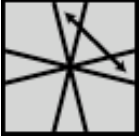

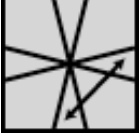

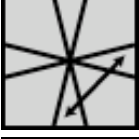
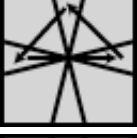
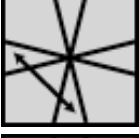


Top view      Bottom view

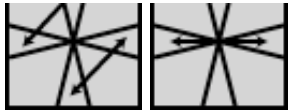
of top layer	of bottom layer	
		(UR UF UL UB) (1,0) / (2,2) / (0,-2) / (3,3) / (1,0) / (4,4) / (0,-2) / (2,2) / (0,-1) / (3,3) / [10 25]
		(UR UB UL UF) / (-3,-3) / (0,1) / (-2,-2) / (0,2) / (-4,-4) / (-1,0) / (-3,-3) / (0,2) / (-2,-2) / (-1,0) / [10 25]
		(UR UB UF UL) (0,-1) / (1,-2) / (-4,0) / (0,3) / (1,0) / (3,-2) / (-4,0) / (-4,0) / (-2,2) / (-1,0) / (0,-3) / (-3,0) / [11 26]

### 1 edge swap in U layer, 1 edge swap in D layer

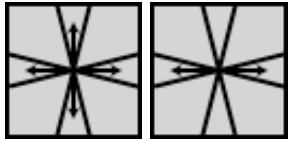
Top view of top layer	Bottom view of bottom layer	
		(UR UF) (DL DF) (1,0) / (0,3) / (-1,-1) / (1,-2) / (-1,0) [4 11]
		(UF UB) (DR DB) (1,0) / (0,-1) / (0,-3) / (5,0) / (-5,0) / (0,3) / (0,1) / (5,0) [7 15]
		(UF UB) (DF DB) (1,0) / (5,-1) / (-5,1) / (5,0) [3 9]

### 2 edge swaps in U layer, 1 edge swap in D layer

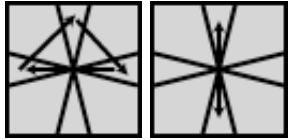
Top view of top layer	Bottom view of bottom layer	
		(UR UF) (UL UB) (DR DF) / (3,0) / (3,3) / (0,-3) / (4,-3) / (4,-2) / (2,-4) / (0,-4) / (3,0) / (-3,-3) / (-3,0) [10 25]
		(UL UR) (UF UB) (DR DB) / (3,3) / (1,4) / (0,-4) / (4,-2) / (2,-4) / (0,-1) / (0,3) / (-3,-3) / (-3,0) / (0,-3) [10 25]
		(UR UL UB) (DR DB) / (3,3) / (1,0) / (-2,0) / (-4,0) / (0,-4) / (0,-4) / (0,-2) / (0,5) / (3,3) / (0,-3) [10 22]
		(UR UB UL) (DB DL) / (-3,-3) / (0,-5) / (0,2) / (0,4) / (0,4) / (4,0) / (2,0) / (-1,0) / (-3,-3) / (0,3) [10 22]
		(UR UF) (UL UB) (DR DL)



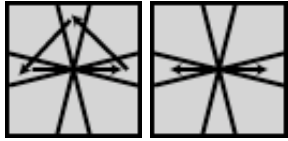
$(3,3) / (1,0) / (-2,4) / (2,-4) / (0,4) / (2,2) / (-3,0) / (-3,-3) / (-3,0)$   
[9|23]



(UL UR) (UF UB) (DR DL)  
 $(3,3) / (1,0) / (-2,4) / (2,-4) / (2,0) / (3,3) / (-3,0) / (3,3) / (3,0)$   
[9|23]



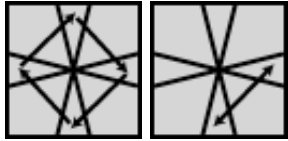
(UR UL UB) (DF DB)  
 $(3,0) / (-3,0) / (3,0) / (0,3) / (1,0) / (0,2) / (4,0) / (0,-4) / (2,0) / (0,5) / (3,3) / (0,-3)$   
[12|25]



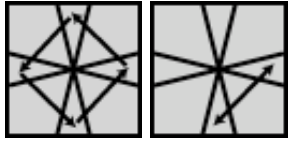
(UR UB UL) (DR DL)  
 $(-3,-3) / (0,-5) / (-2,0) / (0,4) / (-4,0) / (0,-2) / (-1,0) / (0,-3) / (-3,0) / (3,0) / (-3,0) / (0,3)$   
[12|25]

### 3 edge swaps in U layer, 1 edge swap in D layer

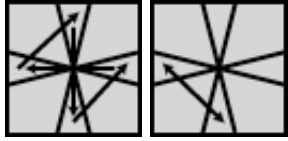
Top view of top layer      Bottom view of bottom layer



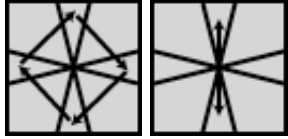
(UR UF UL UB) (DR DB)  
 $(-3,0) / (-2,0) / (5,-1) / (-5,0) / (3,0) / (-3,0) / (0,1) / (-3,0) / (-3,0) / (-1,0)$   
[10|21]



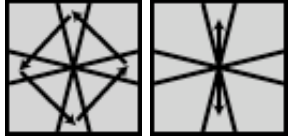
(UR UB UL UF) (DR DB)  
 $(1,0) / (3,0) / (3,0) / (0,-1) / (3,0) / (-3,0) / (5,0) / (-5,1) / (2,0) / (3,0) /$   
[10|21]



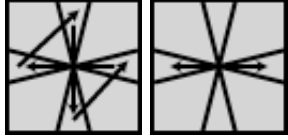
(UR UL UB UF) (DL DB)  
 $(1,0) / (0,3) / (5,5) / (3,0) / (-3,0) / (1,1) / (-3,0) / (-1,6)$   
[7|19]



(UR UF UL UB) (DF DB)  
 $(0,-1) / (1,1) / (-3,0) / (-1,-1) / (-3,0) / (1,1) / (5,0)$   
[6|16]



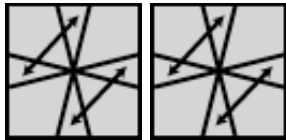
(UR UB UL UF) (DF DB)  
 $(0,-1) / (1,1) / (3,0) / (-1,-1) / (3,0) / (1,1) / (5,0)$   
[6|16]



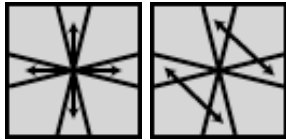
(UR UL UB UF) (DR DL)  
 $(1,0) / (0,3) / (-1,0) / (0,3) / (1,0) / (2,2) / (0,1) / (0,3) / (1,0) / (-3,0) / (-1,0)$   
[10|22]

### 2 edge swaps in U layer, 2 edge swaps in D layer

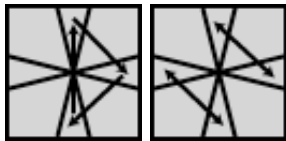
Top view of top layer      Bottom view of bottom layer



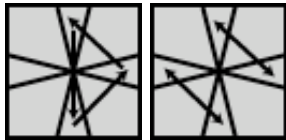
(UR UF) (UL UB) (DL DF) (DR DB)  
 (1,0) / (3,0) / (5,5) / (3,-3) / (1,1) / (-3,0) / (5,6)  
 [6|17]



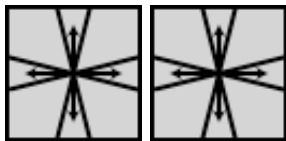
(UL UR) (UF UB) (DR DF) (DL DB)  
 / (3,3) / (0,-3) / (1,-5) / (4,-2) / (-2,1) / (3,3) /  
 [7|18]



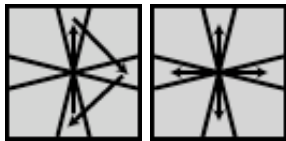
(UR UF UB) (DR DF) (DL DB)  
 (0,-1) / (0,-3) / (1,1) / (5,-3) / (3,0) / (-5,0) / (0,5) / (0,-3) / (-1,0) / (0,-5)  
 [9|21]



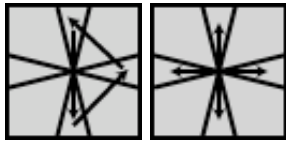
(UR UB UF) (DR DF) (DL DB)  
 (0,-1) / (1,0) / (0,3) / (0,-5) / (5,0) / (-3,0) / (-5,3) / (-1,-1) / (0,3) / (0,-5)  
 [10|21]



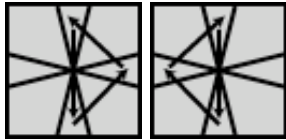
(UL UR) (UF UB) (DL DR) (DF DB)  
 (1,0) / (5,-1) / (3,3) / (1,1) / (3,-3) / (5,0)  
 [5|15]



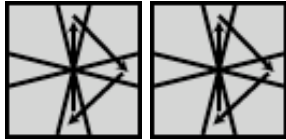
(UL UB UF) (DL DR) (DF DB)  
 (1,0) / (-1,5) / (3,1) / (-3,0) / (0,5) / (0,-5) / (3,0) / (-5,0) / (0,-3) / (5,0)  
 [9|21]



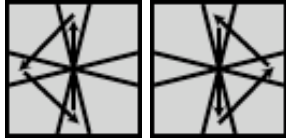
(UR UB UF) (DL DR) (DF DB)  
 (1,0) / (0,3) / (5,0) / (-3,0) / (0,5) / (0,-5) / (3,0) / (-3,-1) / (1,-5) / (5,0)  
 [10|21]



(UR UB UF) (DL DF DB)  
 (1,0) / (5,-1) / (-5,1) / (-3,0) / (0,3) / (-1,-1) / (1,-2) / (-4,0)  
 [7|19]



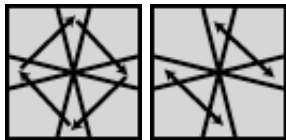
(UR UF UB) (DR DB DF)  
 (1,0) / (-1,5) / (0,-3) / (-5,-5) / (-3,0) / (5,0)  
 [5|13]



(UL UF UB) (DR DF DB)  
 (1,0) / (3,0) / (5,5) / (0,3) / (1,-5) / (5,0)  
 [5|13]

### 3 edge swaps in U layer, 2 edge swaps in D layer

Top view of top layer      Bottom view of bottom layer



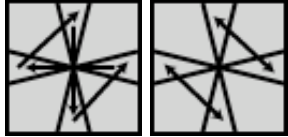
(UR UF UL UB) (DR DF) (DL DB)  
 / (3,3) / (3,0) / (2,2) / (2,0) / (4,-2) / (2,-4) / (2,0) / (5,5) / (0,-3) / (-3,-3) / (0,3)  
 [11|28]



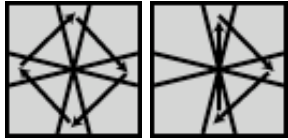
(UR UB UL UF) (DR DF) (DL DB)  
 / (3,3) / (0,3) / (-5,-5) / (-2,0) / (-2,4) / (-4,2) / (-2,0) / (-2,-2) / (-3,0) / (-3,-3) / (0,-3)



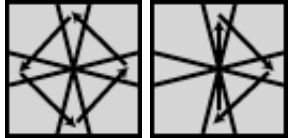
[11|28]



(UR UL UB UF) (DR DF) (DL DB)  
 / (3,3) / (1,-2) / (4,-4) / (2,-4) / (-2,4) / (1,0) / (0,-3) / (3,3) / (3,0) / (3,0)  
 [10|26]



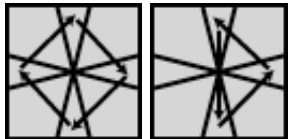
(UR UF UL UB) (DR DB DF)  
 / (3,0) / (0,-3) / (3,0) / (3,0) / (0,5) / (2,-4) / (-4,0) / (0,-4) / (-4,-2) / (1,0) / (-3,-3) / (-3,0)  
 [12|27]



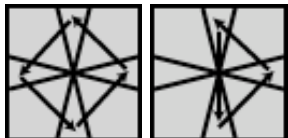
(UR UB UL UF) (DR DB DF)  
 / (3,0) / (-3,-3) / (-3,0) / (0,1) / (4,0) / (-4,2) / (-4,-2) / (0,2) / (4,0) / (-5,0) / (-3,-3) / (0,3)  
 [12|28]



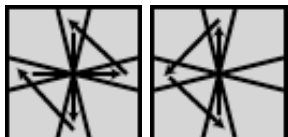
(UR UB UF UL) (DL DB DF)  
 / (0,-3) / (4,0) / (0,-1) / (0,2) / (4,4) / (2,0) / (1,0) / (0,-3) / (3,0) / (-4,0) / (3,0) / (0,-3)  
 [12|26]



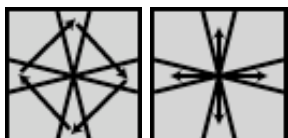
(UR UF UL UB) (DR DF DB)  
 / (3,3) / (5,0) / (-4,0) / (0,-2) / (4,2) / (4,-2) / (-4,0) / (0,-1) / (3,0) / (3,3) / (-3,0) / (0,-3)  
 [12|28]



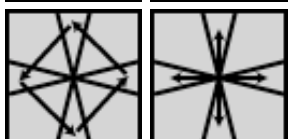
(UR UB UL UF) (DR DF DB)  
 / (3,3) / (-1,0) / (4,2) / (0,4) / (4,0) / (-2,4) / (0,-5) / (-3,0) / (-3,0) / (0,3) / (-3,0) / (3,0)  
 [12|27]



(UR UB UF UL) (DL DB DF)  
 / (-3,0) / (4,0) / (-3,0) / (0,3) / (-1,0) / (-2,0) / (-4,-4) / (0,-2) / (0,1) / (-4,0) / (0,3) / (0,3)  
 [12|26]



(UR UF UL UB) (DR DF DB)  
 / (3,3) / (1,-2) / (2,2) / (2,0) / (2,-4) / (4,-2) / (0,-5) / (3,3) / (3,0)  
 [9|24]



(UR UB UL UF) (DR DF DB)  
 / (-3,-3) / (0,5) / (-4,2) / (-2,4) / (-2,0) / (-2,-2) / (-1,2) / (-3,-3) / (-3,0)  
 [9|24]



(UR UF UB UL) (DL DR) (DF DB)  
 / (3,0) / (0,-3) / (3,0) / (3,0) / (2,-1) / (0,2) / (-2,4) / (2,-4) / (2,0) / (-1,2) / (-3,-3) / (6,0)  
 [12|29]



# Lars Vandenberg's CubeZone

Belgium's No. 1 speedcuber

[Algorithms](#) [ZB F2L](#) | [OLL](#) | [PLL](#) | [COLL](#) **Methods** [Square-1](#) **Tools** [ImageCube](#) | [ImageRevenge](#) **Media** [News articles](#) | [Videos](#) **Miscellaneous** [About me](#)  
[Overview](#) | [Notation](#) | [Step 1](#) | [Step 2](#) | [Step 3](#) | [Step 4](#) | [Step 5](#) | **Step 6**

## Square-1 solution method - Step 6

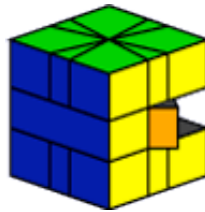
Correct middle layer and swap layers if necessary

Swap layers



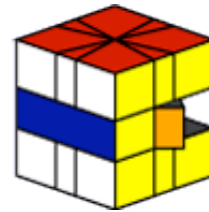
$(1,0) / (6,6) / (-1,0)$

Flip middle layer



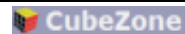
$/ (6,0) / (6,0) / (6,0)$

Swap layers and flip middle layer



$/ (6,0) / (0,6) / (-1,-5)$

This page is maintained by [Lars Vandenberg](#)



Last update on 4th September 2005