IMPROVER METHOD – PT. 4

MICHAEL GEORGE

WCA ID: 2015GEOR02

PERMUTATION OF THE LAST LAYER (**PLL**)

"Beginner" method used 2 simple algorithms

- 16-move combination of **Anti-Sune** and **Niklas** for corner permutation (**CPLL**)
- 17-move combination of **Anti-Sune** and its left-handed mirror for edge permutation (**EPLL**)

"Improver" method will use 3 shorter but less intuitive algorithms

- 11-move algorithm for corner permutation (**Jb-Perm**) – **R U L** moves
- 11-move algorithms for edge permutation (**Ua-Perm** and **Ub-Perm**) – **R U** moves

The new algorithms will be slightly trickier to learn but well worth the effort!

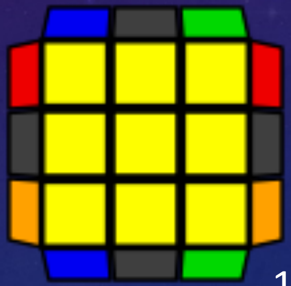
- The combination of **Anti-Sune** and **Niklas** will be shortened to 11 moves – optimised **Jb-Perm**
- The **Ua-Perm** and **Ub-Perm** algorithms will be speed-optimised to 11 moves – **R U** only

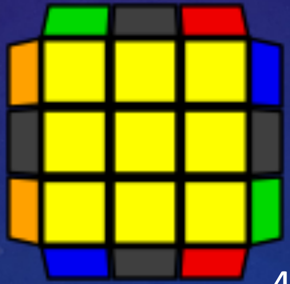
CORNER PERMUTATION OF THE LAST LAYER (CPLL)

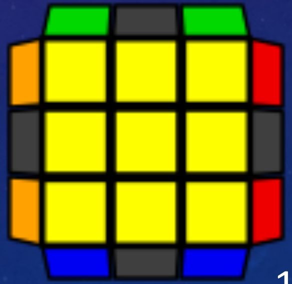
There are 3 possible cases during **CPLL** including the "**solved**" case

The cases that need to be solved are the "**diagonal corner swap**" and the "**adjacent corner swap**"

The "**adjacent**" corner swap is the most common **CPLL** as it occurs in **4/6** solves


1/6


4/6


1/6

CPLL ALGORITHM

To change the permutation of corners you will use a mixture of **R U L** moves

- The **CPLL** algorithm that will be used is **R U2 R’ U’ R U2 L’ U R’ U’ L**
    - It may not be immediately obvious but it is actually the beginner algorithm after cancellations – see square brackets
    - It starts with the **Anti-Sune** algorithm that was used during **OCLL** – **R U2 R’ U’ R [U’ R’]**
    - It ends with the **Niklas** algorithm which manipulates two **F2L pairs** – **[R U’] L’ U R’ U’ L [U]**
    - The combination of these two algorithms is the **Jb-Perm** which swaps 2 LL corners and 2 LL edges

- Explanation of the algorithm
    - The **Anti-Sune** algorithm disorients 3 of the LL corners and permutes 3 of the LL edges
    - The **Niklas** algorithm re-orients the LL corners and changes their permutation
    - The combined effect of these two algorithms is to maintain the **OLL** but change the **PLL**

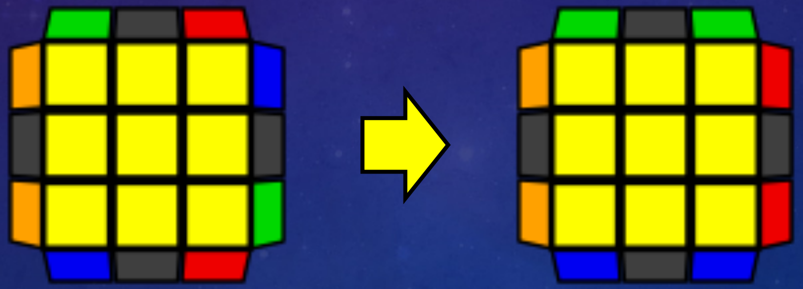
ADJACENT CORNER SWAP

The “**adjacent**” corner swap can be recognised by the “**headlights**” on one side (shown in orange below)

Approach: “**Adjacent corner swap**” -> “**solved**”

Setup: Ensure the “**Headlights**” are on the left before executing the **CPLL** algorithm

Algorithm: **R U2 R’ U’ R U2 L’ U R’ U’ L**



Adjacent Swap **Solved**

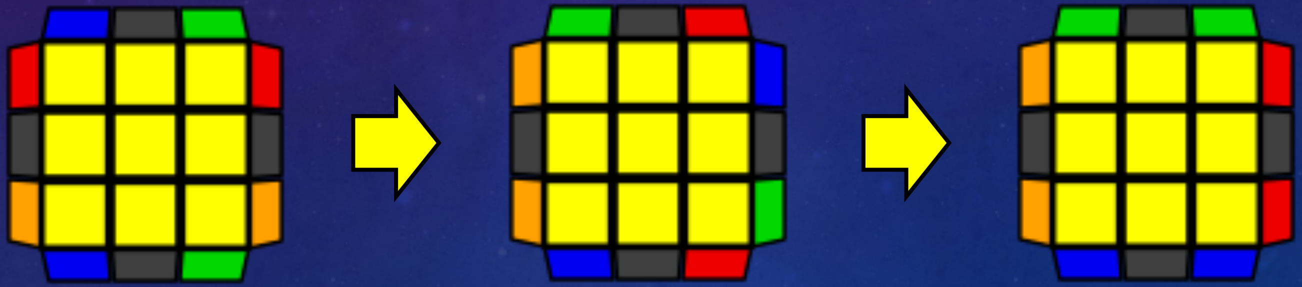
DIAGONAL CORNER SWAP

The “**diagonal**” corner swap can be recognised by the absence of “**headlights**”

Approach: “**Diagonal corner swap**” -> “**adjacent corner swap**” -> “**solved**”

Setup: Ensure the “**Headlights**” are on the left before executing the ”**adjacent corner swap**”

Algorithm: **(R U2 R’ U’ R U2 L’ U R’ U’ L) U2 (R U2 R’ U’ R U2 L’ U R’ U’ L)**



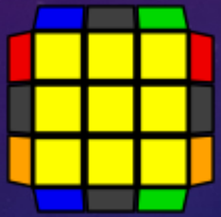
CPLL EFFICIENCY

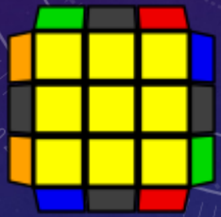
"Beginner" – 1 algorithm – **(R U2 R' U2') (U R U' R') (R U' L' U) (R' U' L U)**

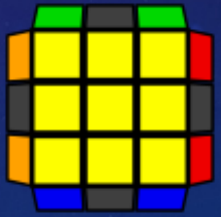
• **Adjacent Corner Swap**	**16** moves, **2** looks
• **Diagonal Corner Swap**	**33** moves, **4** looks
• **"Weighted Average"**	**16.2** moves, **2.2** looks

"Improver" – 1 algorithm – **R U2 R' U' R U2 L' U R' U' L**

• **Adjacent Corner Swap**	**11** moves, **1** look
• **Diagonal Corner Swap**	**23** moves, **2** looks
• **"Weighted Average"**	**11.2** moves, **1.2** looks


Diagonal

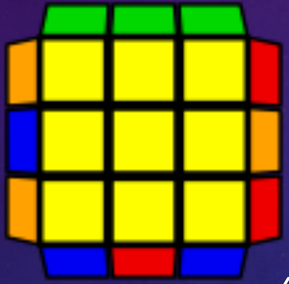

Adjacent

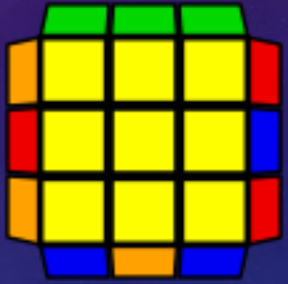

Solved

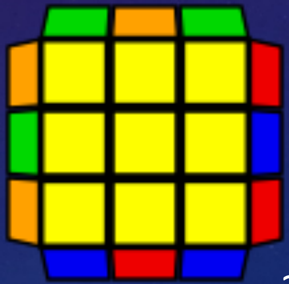
EDGE PERMUTATION OF THE LAST LAYER (EPLL)

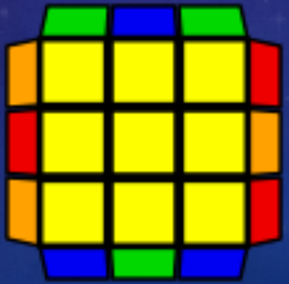
There are 5 possible cases during **EPLL** including the "**solved**" case

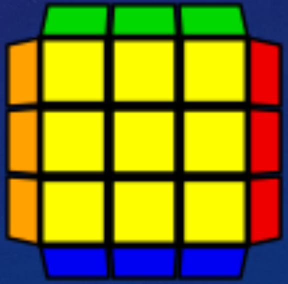
4 misplaced edges require the most effort to solve when you only know a couple of algorithms


4/12


4/12


2/12


1/12


1/12

EPLL ALGORITHMS

To change the permutation of edges we will only use **R U** moves

- The first **EPLL** algorithm that will be used is **R U’ R U R U R U’ R’ U’ R2**
    - It is a conjugation of an algorithm that can be used during **F2L** – **R U’ (R U R U R U’ R’ U’ R’ U’) U R’**
    - The **R U’** is referred to as a “**setup**” move and the **U R’** will undo the setup
    - Applying cancellations results in the **Ua-Perm** which is a counter-clockwise “**3-cycle**” of LL edges

- The second **EPLL** algorithm that will be used is **R2 U R U R’ U’ R’ U’ R’ U R’**
    - It is a conjugation of an algorithm that can be used during **F2L** – **R U’ (U R U R U R’ U’ R’ U’ R’) U R’**
    - The **R U’** is referred to as a “**setup**” move and the **U R’** will undo the setup
    - Applying cancellations results in the **Ub-Perm** which is a clockwise “**3-cycle**” of LL edges

Note how the two algorithms are both an “**inverse**” of each other

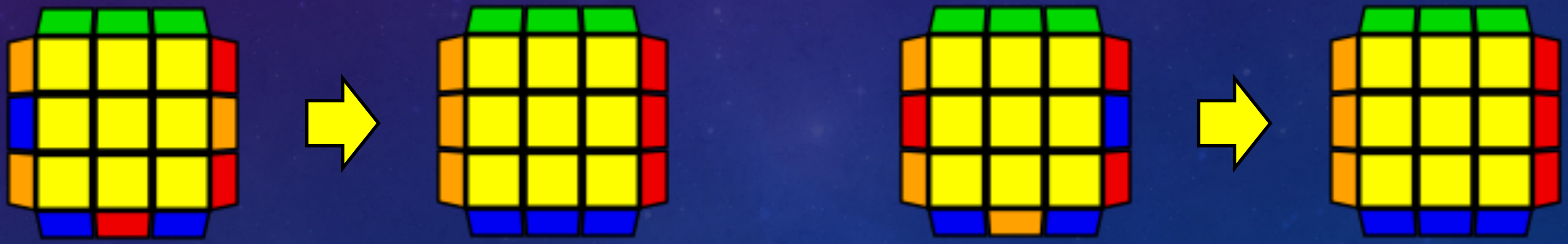
3 MISPLACED EDGES

There are 2 **EPLL** cases with 3 misplaced edges and they can be recognised by the “**bar**” (green below)

Approach: “**Ua-Perm**” or “**Ub-Perm**” -> “**Solved**”

Setup: Ensure the ”**Bar**” is at the **back** before executing the **EPLL** algorithm

Algorithm: **R U’ R U R U R U’ R’ U’ R2** or **R2 U R U R’ U’ R’ U’ R’ U R’** – depending on the case



Ua-Perm **Solved** **Ub-Perm** **Solved**

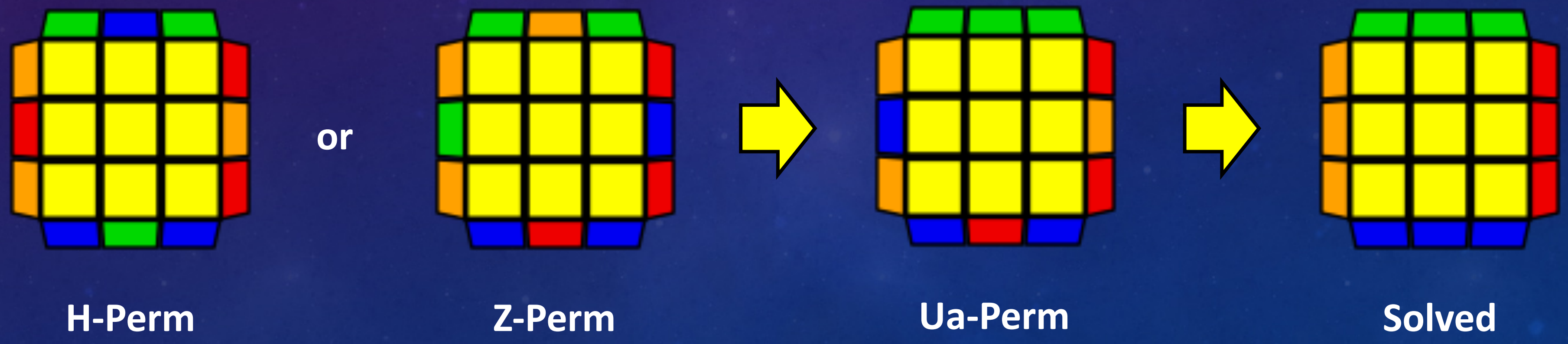
4 MISPLACED EDGES

There are 2 **EPLL** cases with 4 misplaced edges and they take the most effort to solve

Approach: "**Z-Perm**" or "**H-Perm**" -> "**Ub-Perm**" -> "**Solved**"

Setup: Avoid the "**Ua-Perm**" by setting up the Z-Perm correctly before executing the **EPLL** algorithm

Algorithm: **R U' R U R U R U' R' U' R2** – twice, remembering to **AUF** prior to execution



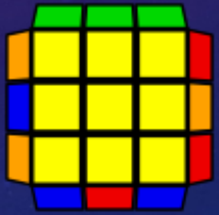
EPLL EFFICIENCY

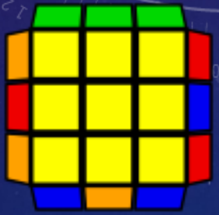
"Beginner" – 1 algorithm – **(R U2 R' U2') (U R U' R') U (L' U2 L U2') (U' L' U L)**

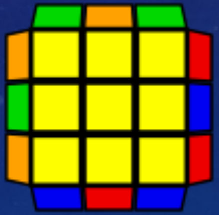
• **3 Misplaced Edges** – **Ua** or **Ub**	**17** or **35** moves, **2** or **4** looks
• **4 Misplaced Edges** – **Z** or **H**	**34** or **35** moves, **4** looks
• **"Weighted Average"**	**25.9** moves, **3.1** looks

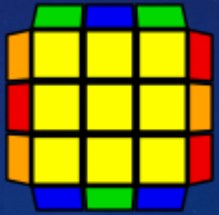
"Improver" – 2 algorithms – **R U' R U R U R U' R' U' R2** and **R2 U R U R' U' R' U' R' U R**

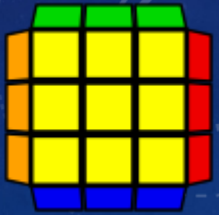
• **3 Misplaced Edges** – **Ua** or **Ub**	**11** moves, **1** look
• **4 Misplaced Edges** – **Z** or **H**	**23** moves, **2** looks
• **"Weighted Average"**	**13.1** moves, **1.3** looks


Ua-Perm


Ub-Perm


Z-Perm


H-Perm


Solved

PLL EFFICIENCY

"Beginner" – 2 algorithms – beginner **Jb-Perm** + beginner **Ub-Perm**

- **"Worst Case"** **68** moves, **8** looks
- **"Weighted Average"** **42.1** moves, **5.3** looks

"Improver" – 3 algorithms – **Jb-Perm** + **Ua-Perm** + **Ub-Perm**

- **"Worst Case"** **46** moves, **4** looks
- **"Weighted Average"** **24.3** moves, **2.4** looks

Comparison

- **"Worst Case"** "Improver" method saves **22** moves, **4** looks –> **32.4%** saving
- **"Weighted Average"** "Improver" method saves **17.8** moves, **3** looks –> **42.3%** saving

CONGRATULATIONS!