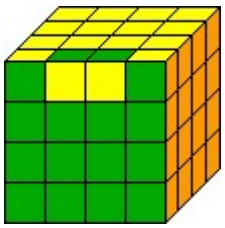
Hier habt ihr die Algorithmen für die 2 möglichen Sonderfälle (parities) beim **4x4-Cube**.
Die Algorithmen sind auch für alle größeren Cubes gültig und anwendbar!

Parity-Algorithmen

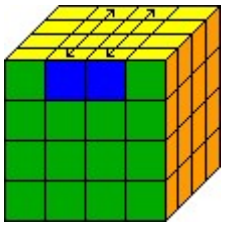
Bei Big Cubes können während des Lösens wie einen 3er (nach der Reduction) 2 Sonderfälle auftreten. Mit folgenden Algorithmen lassen sie sich lösen. Kleinbuchstaben stehen hier für innere Kanten, Großbuchstaben für äußere Kanten!

OLL-Parity



(Anwendung bei OLL) Hier werden die 2 Einzel-Kantensteine vertauscht, wodurch die "Gesamtkante" wieder richtig gekippt wird. Man kann bei dem OLL-Parity statt den inneren Ebenen (r) auch beide Ebenen drehen (R+r = Rw), was drehfreudiger ist und beim normalen Anwenden während des OLL keine nachteiligen Veränderungen hervorruft.
r2 B2 U2 l U2 r' U2 r U2 F2 r F2 l' B2 r2

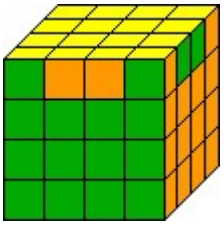
PLL-Parity



(Anwendung bei PLL) Beim PLL-Parity lässt sich der Cube durch eine normale Permutation nicht lösen. Der Algorithmus vertauscht 2 gegenüberliegende Kantenpaare, wodurch eine normale Permutation möglich wird.
(Das Bild ist eigentlich falsch, die Pfeile müssten sich kreuzen)
r2 U2 r2 Uu2 r2 u2

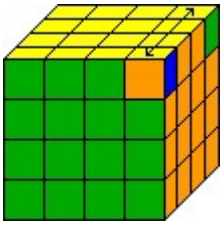
Verschiedenes Auftreten des PLL-Parities

PLL-Parity



Achtung: Kein Z-Perm! Alle anderen Seiten sind gelöst. Bei diesem Fall kann man den PLL-Parity-Algorithmus nicht direkt anwenden, man muss zunächst einen Setup-Move machen, wie z.B. **F U' F'**, und kann dann den oben angegebenen Algorithmus verwenden.

PLL-Parity



Dieser Fall wird irrtümlicherweise als eigener Parity angesehen Dabei handelt es sich auch um einen PLL-Parity. Das sieht man allerdings erst, wenn man mit einem Algorithmus, z.B. T-Perm, die Ecken richtig angeordnet hat. Auch hier lässt sich der oben angegebene Algorithmus verwenden.