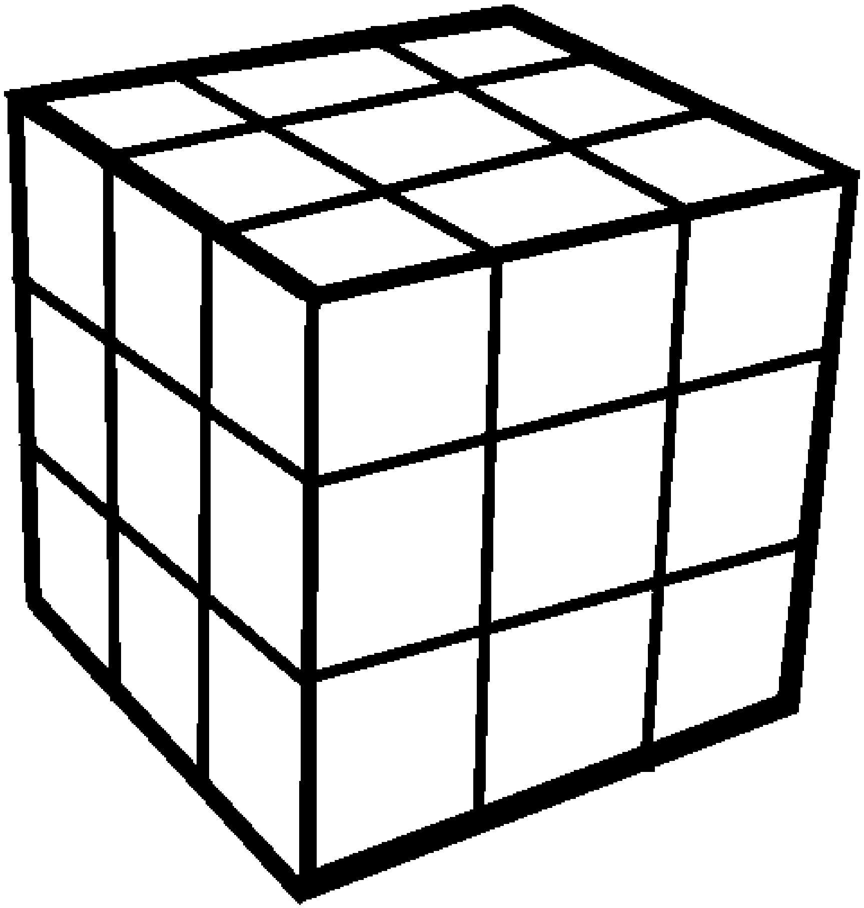
Présentation de la méthode de résolution des rubik's cube de la forme NxNxN



Objectifs

Le but de ce projet est de donner un algorithme de résolution pratique d'un rubik's cube en trois dimensions de forme cubique et comportant N découpages par arrête et dont les faces soient unicolores (sans motif).

Prérequis nécessaires

Notions élementaires de la théorie des groupes

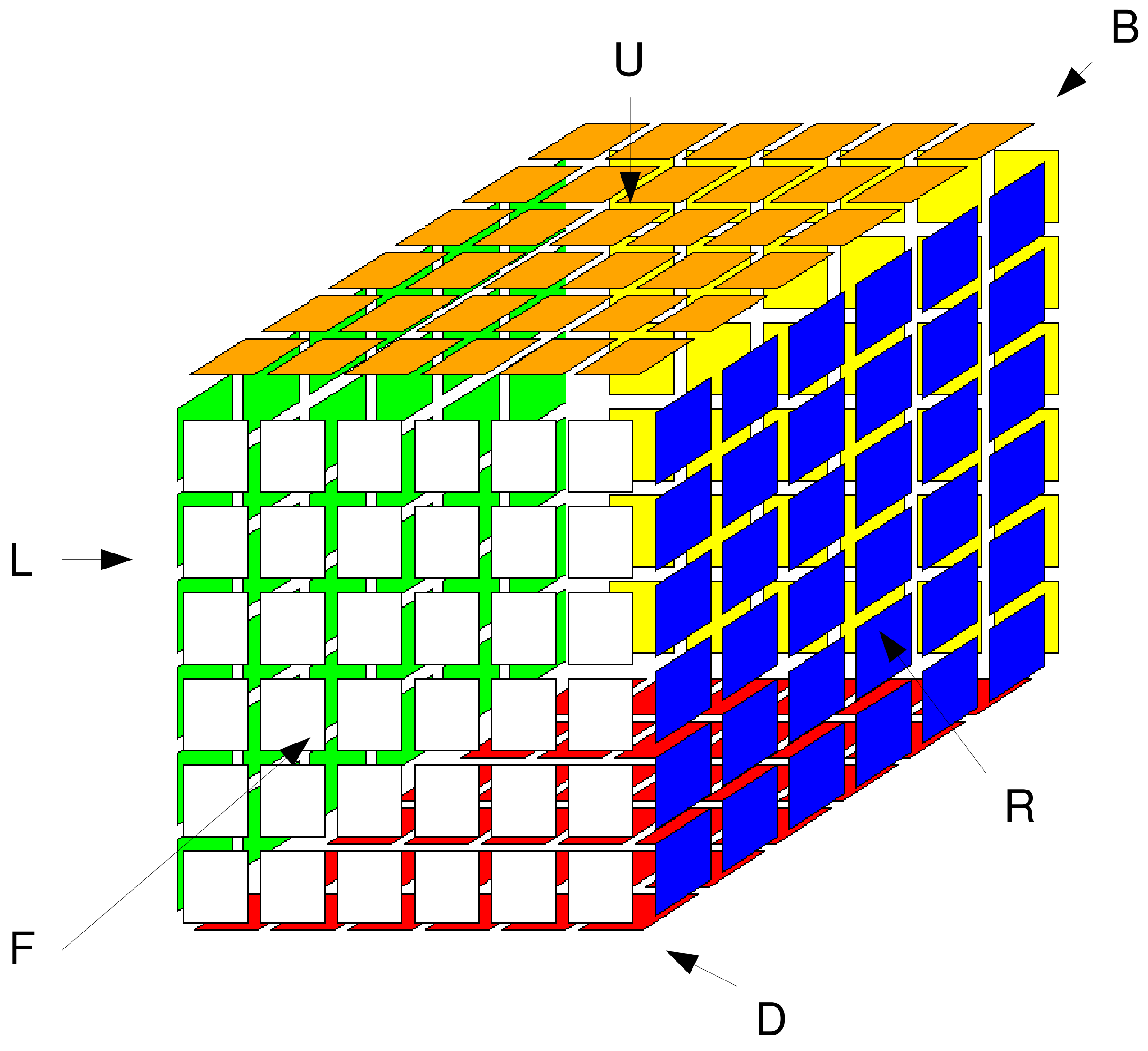
Notions élémentaires à propos des permutations

Notions élémentaires d'algorithmique

Savoir résoudre les Rubik's cube 3x3x3 et 4x4x4

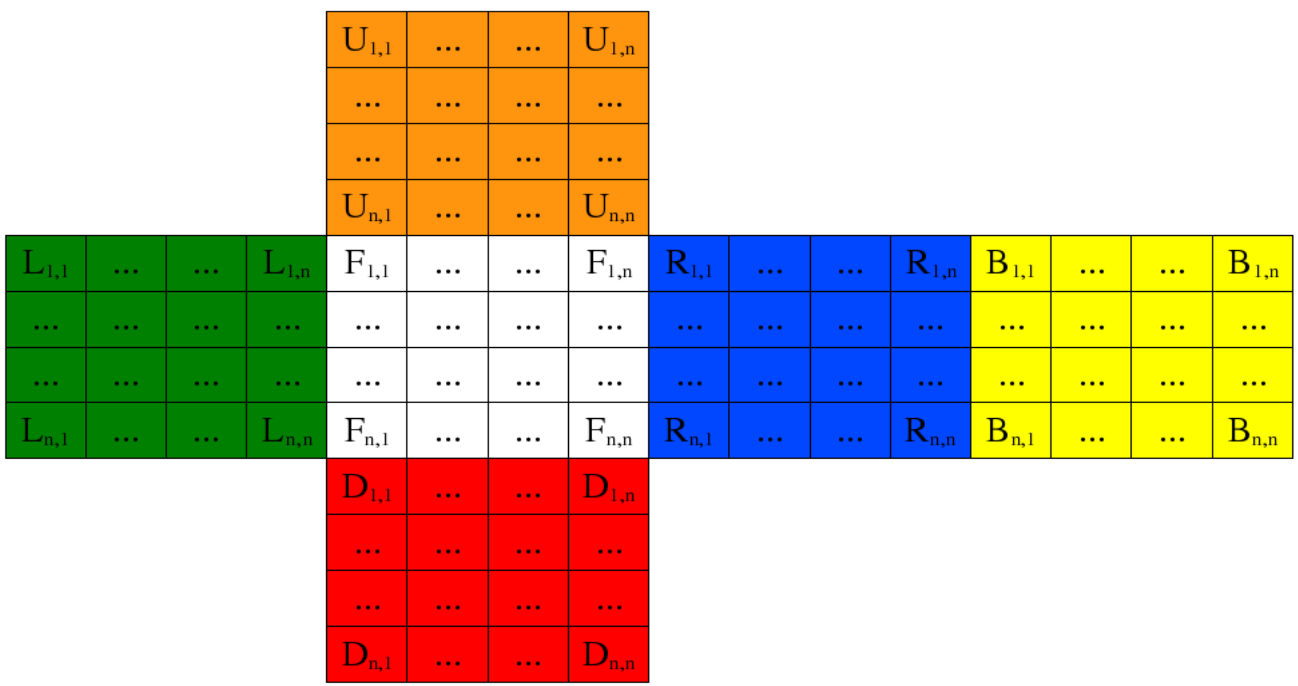
Etudes des configurations impossibles du Rubik's cube 3x3x3 et 4x4x4

Définitions et notations



Soit M = {F,R,B,D,L,U} et E_n = [|1,n|].

On représente mathématiquement le cube par une fonction « cube » de $M \times E_n^2$ dans l'ensemble des couleurs du cube en choisissant pour le cube parfaitement fait la fonction défini par le graphe suivant :



On définit S_n = {ensemble des permutations de $M \times E_n^2$}.

On définit les fonctions :

$$rot : [\![1, n-1]\!] \times M \to S_n$$

$$(p, X) \mapsto \begin{cases} si\ p = 1 & \left(\bigcirc_{i=1}^{\frac{n}{2}} \left(\bigcirc_{j=i}^{n-i} ((X, i, j)(X, j, n+1-i)(X, n+1-i, n+1-j)(X, n+1-j, i)) \right) \right) \\ & \circ \left(\bigcirc_{i=1}^{n} \left(\left(b(X), \sigma_{X, b(X)}(i, 1) \right) \left(g(X), \sigma_{X, g(X)}(i, 1) \right) \left(h(X), \sigma_{X, h(X)}(i, 1) \right) \left(d(X), \sigma_{X, d(X)}(i, 1) \right) \right) \right) \\ si\ p > 1 & \bigcirc_{i=1}^{\frac{n}{2}} \left(\bigcirc_{j=i}^{n-i} ((X, i, j)(X, j, n+1-i)(X, n+1-i, n+1-j)(X, n+1-j, i)) \right) \end{cases}$$

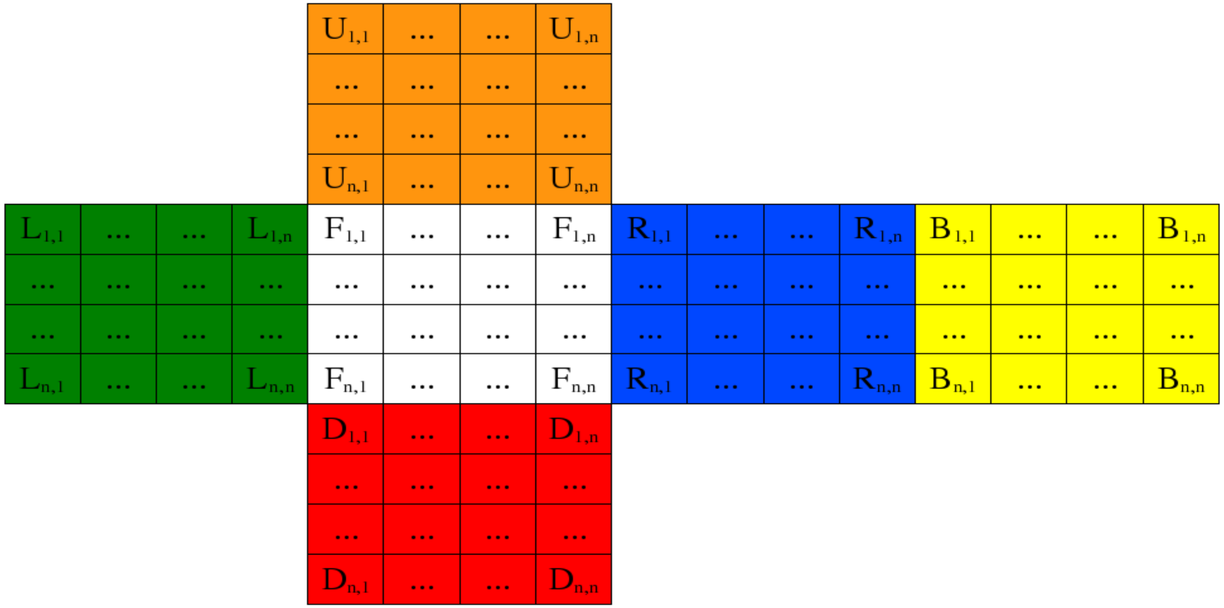
$$\text{et} \quad rot_k(X) = \bigcirc_{i=1}^{k} rot(i, X)$$

On définit C_n =

$$\{\ s \in S_n \mid \exists u \in ([\![1,n-1]\!] \times M)^{\mathbb{N}} ; \exists p \in \mathbb{N} ; s = \bigcirc_{i=1}^{p} rot(u_i)\ \}$$

$$sigma_{X,Y}(i,j) : E_n \text{->} E_n$$

$$(i,j) \text{->} les\ coordonnees\ reorientees$$



Les fonctions g,d,h,b sont des fonctions de M dans M qui à un nom de face associe le nom de la face de gauche, de droite, d'en haut ou d'en bas d'après l'orientation du dessin ci-dessus.

On définit la fonction de symétrie suivante :

$$sym_{RL} : S_n \text{->} S_n$$
$$X \text{->} GXG$$

Avec G dans S_n défini par :

$$G = \underset{X \in \{U,F,D,B\}}{\circ} \left[\underset{i=1}{\overset{E(\frac{n}{2})}{\circ}} \left(\underset{j=1}{\overset{n}{\circ}} ((X,j,i)(X,j,n+1-i)) \right) \right]$$
$$\circ \left[\underset{i=1}{\overset{n}{\circ}} \left(\underset{j=1}{\overset{n}{\circ}} ((L,j,i)(R,j,n+1-i)) \right) \right]$$

La fonction de sym_{RL} restreinte à C_n définit un automorphisme :

<u>Dém</u> : On vérifie facilement que :

sym_{RL} : | Id -> Id (car G^2=Id) | rot(i,B) -> rot^{-1}(i,B)
| rot(i,U) -> rot^{-1}(i,U) | rot(i,L) -> rot^{-1}(i,R)
| rot(i,D) -> rot^{-1}(i,D) | rot(i,R) -> rot^{-1}(i,L)
| rot(i,F) -> rot^{-1}(i,F)

La fonction est bijective car involutive .

Soit V dans C_n, on a :

$$sym_{RL}(V)=sym_{RL}(\mathop{\circ}_{i=1}^{p} rot(u_i))=G\circ(\mathop{\circ}_{i=1}^{p} rot(u_i))\circ G=\mathop{\circ}_{i=1}^{p}(G\circ rot(u_i)\circ G)=\mathop{\circ}_{i=1}^{p}(sym_{RL}(rot(u_i)))$$

Donc la fonction va de C_n dans C_n.

On a aussi $sym_{RL}(V^{-1}) = GV^{-1}G = (GVG)^{-1} = sym_{RL}^{-1}(V)$

Donc sym_{RL} est un automorphisme.

De la même façon on défini les fonctions r_X

$$r_X : C_n \text{->} C_n$$
$$Y \text{->} T_X^{-1} Y T_X$$

Avec T_X défini par :

On verifie de même que r_X est un automorphisme.

On définit pour finir les fonctions suivantes :

$sym_{UD} = sym_{RL} \circ r_F$
$sym_{BF} = sym_{RL} \circ r_U$

L'existence de ces fonctions permet par la suite d'éviter d'être obligé de refaire 48 fois le même calcul de permutation car elles nous assurent que si on tourne le cube avant de faire des mouvements ou si l'on fait les mouvements symétriques le résultat sera le même.

Etapes de la méthode (dans les grandes lignes) :

I – Les contours :

1 - Contour de la première face

2 - Contour des faces adjacentes

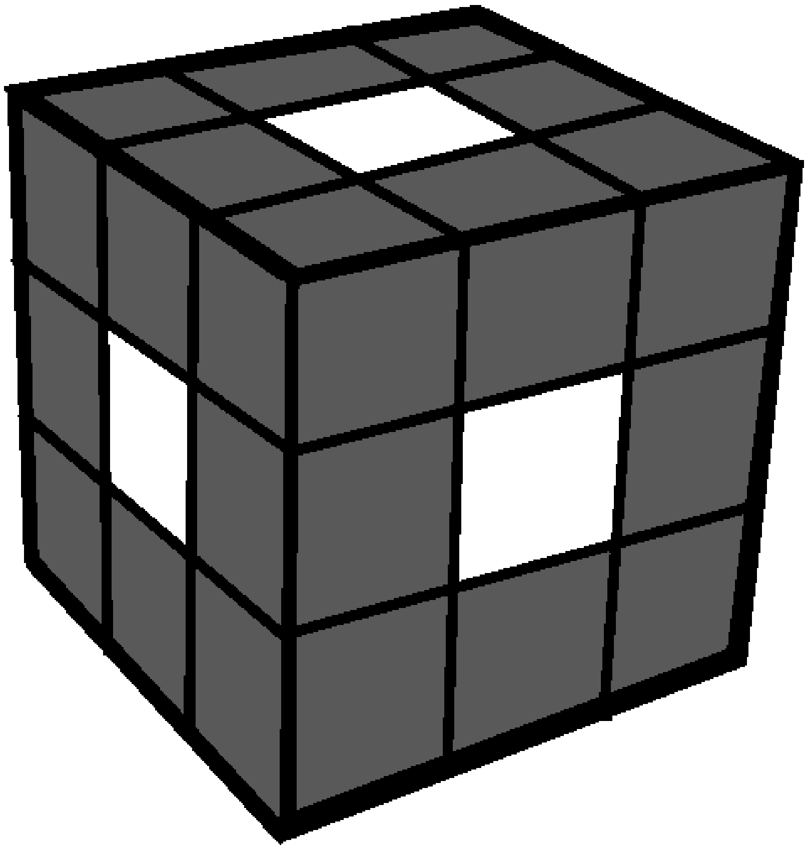
3 - Contour de la dernière face

II – L'intérieur des faces

1 - Les trois premières faces

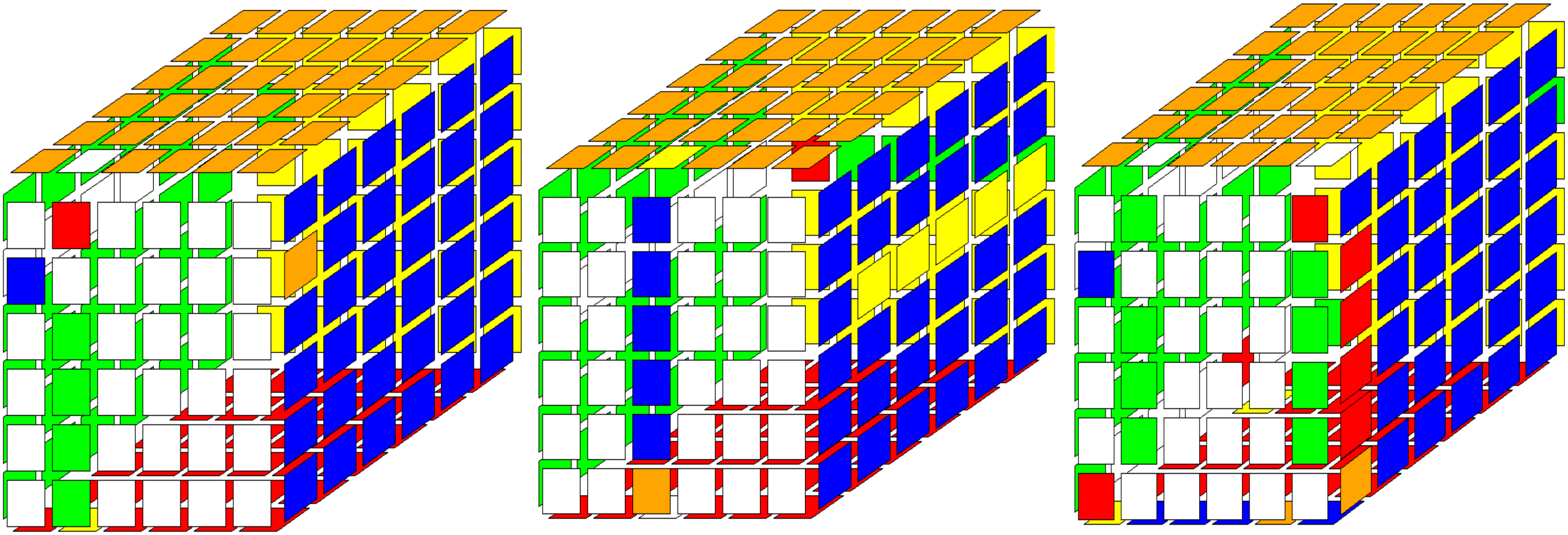
2 – Les trois dernières faces

Les contours



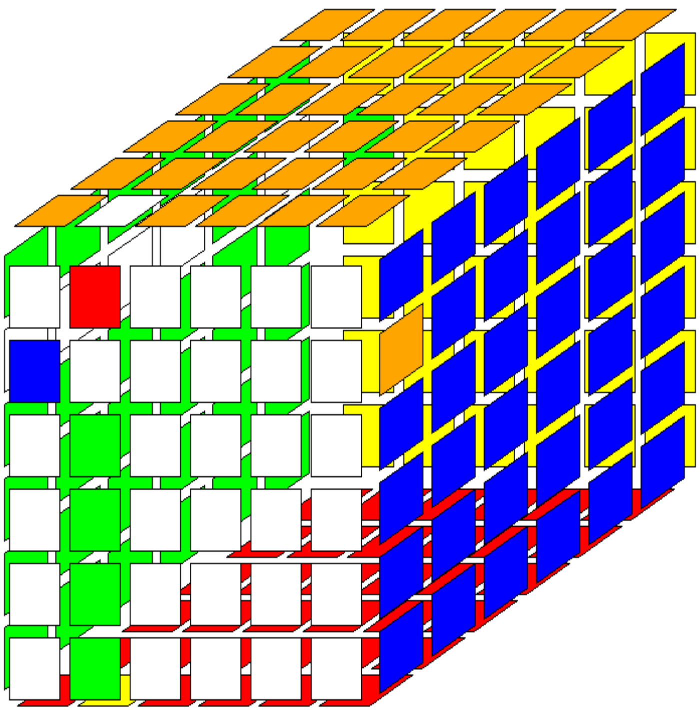
Les arrêtes de la première face

3 cas :



Pourquoi seulement ces cas ?

Cas 1



$$A_1 = rot(1,F)rot^{-1}(i,U)rot^{-1}(1,F)rot(i,U)$$

inverse

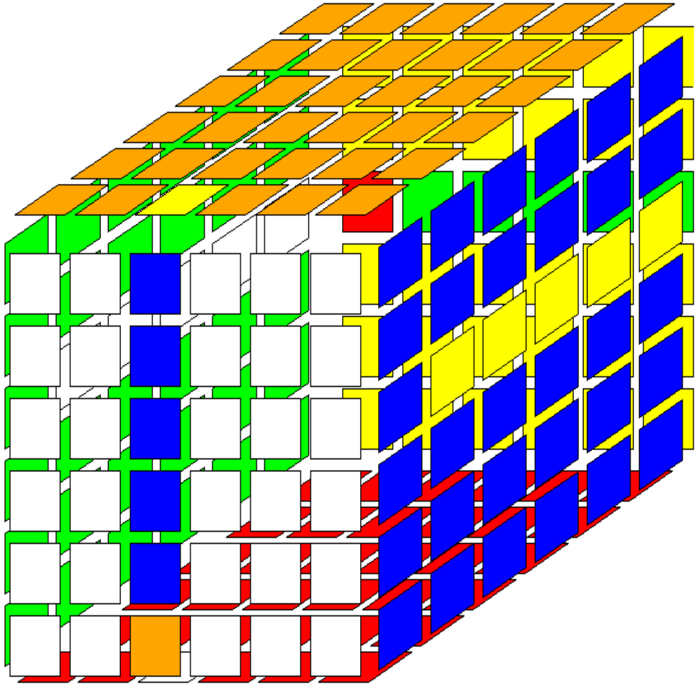
$$A_1^{-1} = rot^{-1}(i,U)rot(1,F)rot(i,U)rot^{-1}(1,F)$$

$$= ((h(F),\sigma_{F,h(F)}(1,n+1-i))(d(F),\sigma_{F,d(F)}(1,n+1-i)))$$

$$o\ ((F,1,n)(F,i,n))\ o\ P$$

avec P une permutation qui laisse invariants les (U,i,j)

Cas 2



$A_2 = rot(1,F)rot(i,U)rot^{-1}(1,F)$

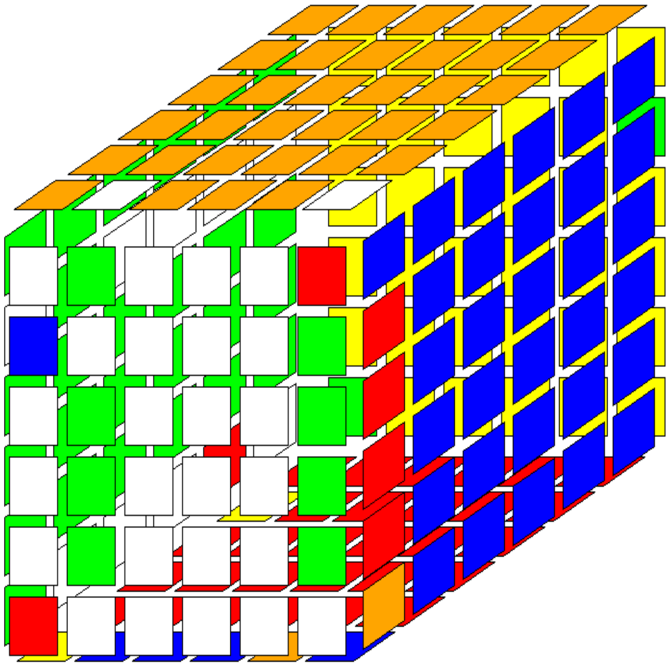
inverse

$A_2^{-1} = rot(1,F)rot^{-1}(i,U)rot^{-1}(1,F)$

$= ((h(F),\sigma_{F,h(F)}(1,n+1-i))(F,n,i)) \text{ o } ((b(F),\sigma_{F,b(F)}(1,i))(F,1,i)) \text{ o } P$

avec P une permutation qui laisse invariants les (U,i,j)

Cas 3



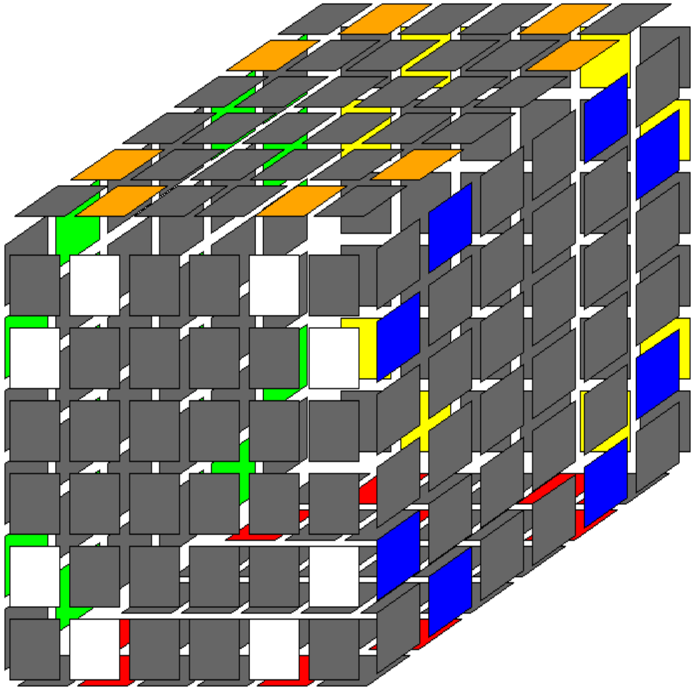
$A_3 = rot(1,F)rot^{-1}(i,U)rot^{-1}(1,F)rot(i,U)rot(1,F)$
$rot(1,D)rot^{-1}(1,F)rot^{-1}(1,D)$

inverse

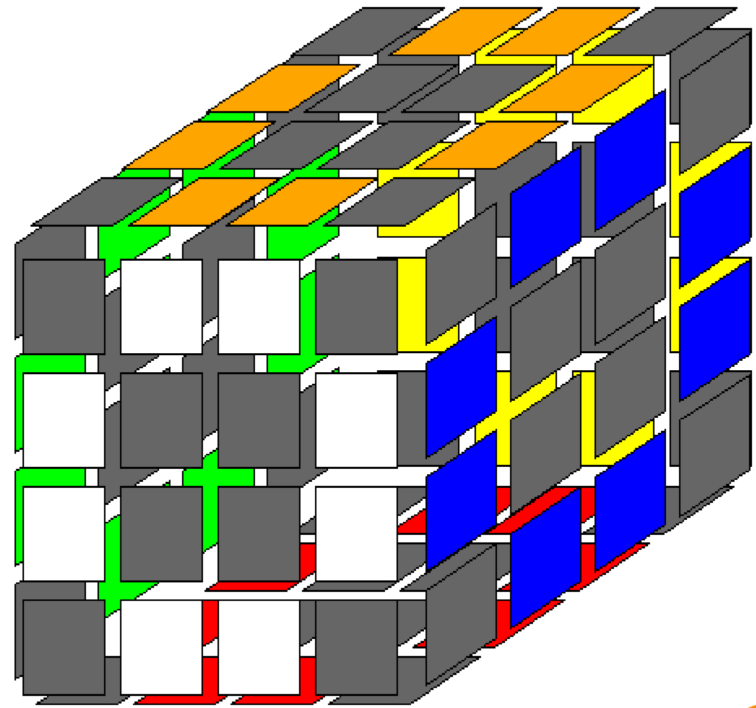
$A_3^{-1} = rot(1,D)rot(1,F)rot^{-1}(1,D)rot^{-1}(1,F)rot^{-1}(i,U)rot(1,F)$
$rot(i,U)rot^{-1}(1,F)$
$= ((h(F),\sigma_{F,h(F)}(1,n+1-i))(b(F),\sigma_{F,b(F)}(1,n+1-i)))$
$o\ ((F,1,i)(F,n,n+1-i))\ o\ P$
avec P une permutation qui laisse invariantes les arrêtes de U

Isomorphisme entre les deux groupes suivants :

$S_{a,N,i}$ = groupe des mouvements qui modifient les morceaux d'arrête de profondeur i par rapport à une face sur un NxNxN



S_a = groupe des mouvements qui modifient les arrêtes du 4x4x4



L'isomorphisme à appliquer n'est autre qu'une permutation de S_n qui renumérote les facettes du cube.
Or en étudiant les 4x4x4 on montre qu'il n'y a que ces cas (les autres étants des configurations impossibles du cube).

Utilisation

Soit X le nom de la face que l'on veut refaire en premier. Il y a exactement 4(N-2) pièces dans les arrêtes d'une face du Rubik's Cube. On numérote celles de X de 1 à 4(N-2). On applique ensuite l'algorithme suivant :

Pour i allant de 1 à 4(N-2) on regarde si la i-ème pièce de X est bien placée, si c'est le cas, on passe à i+1. Sinon :

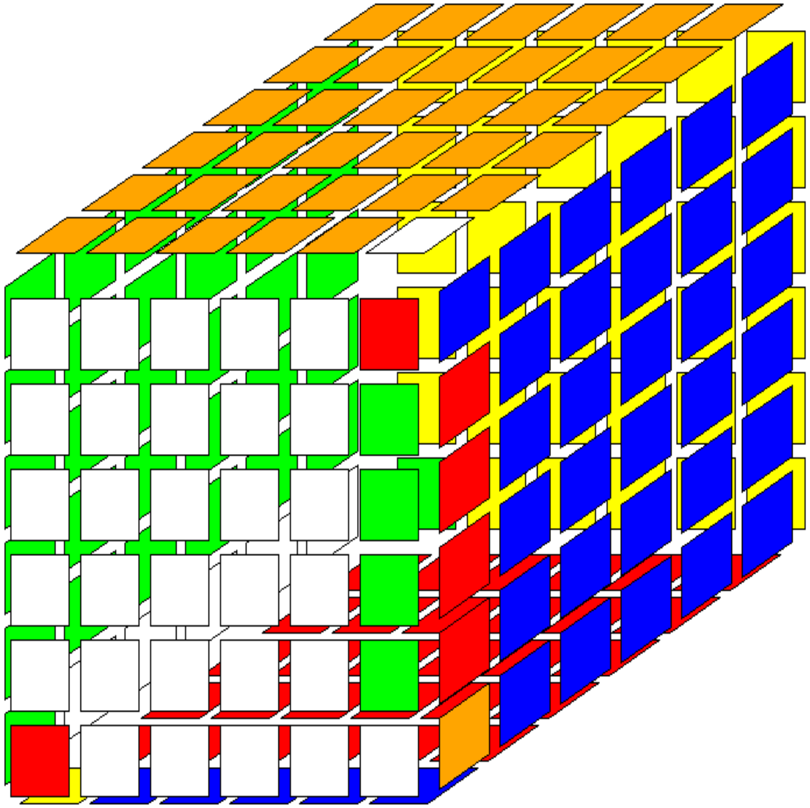
- soit la pièce est à la place d'une autre pièce j (avec j > i) dans ce cas, on la retire en utilisant un des A_k (*) puis on la replace en utilisant un A_k^{-1} (*).
- soit on peut utiliser directement un A_k^{-1} (*).

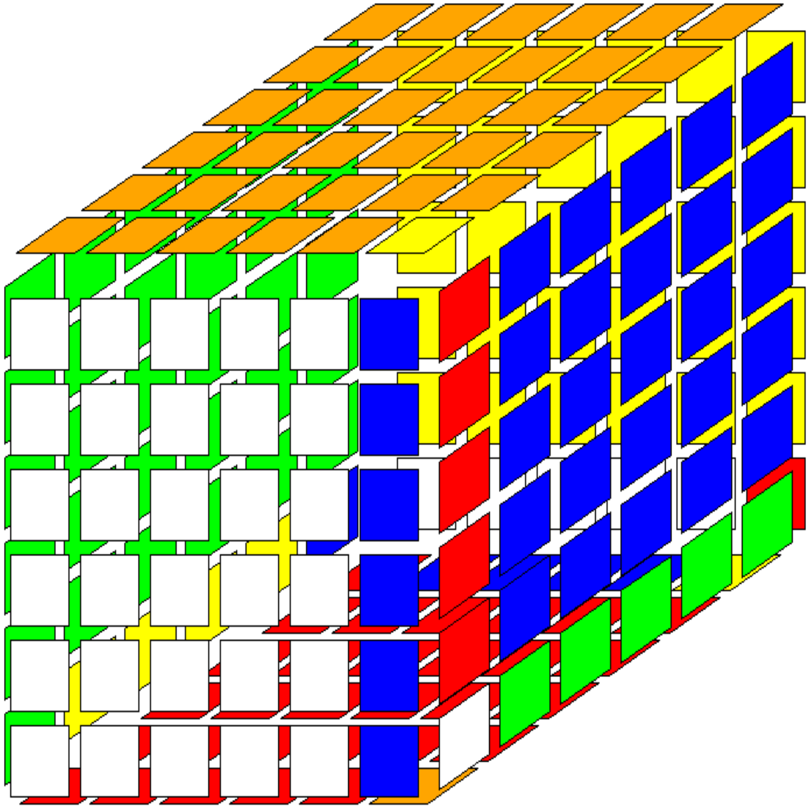
Et comme les A_k ne déplacent pas les autres pièces de X l'algorithme termine correctement quand i atteint 4(N-2).

(*) : en appliquant à A_k une (ou plusieurs) fonction(s) sym ou r si nécessaire

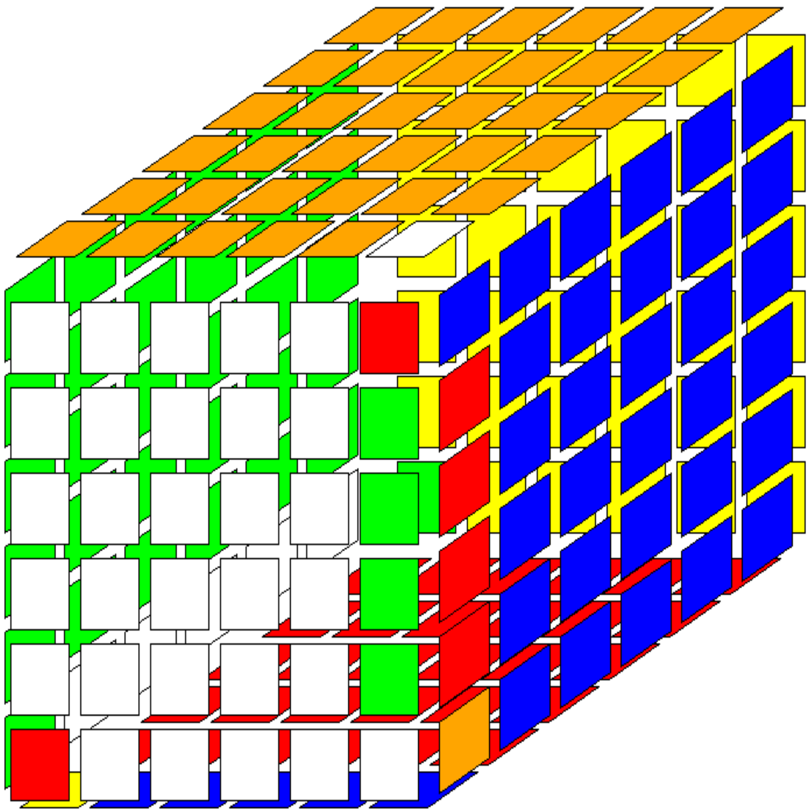
Coins de la première face

2 cas :





Cas 1



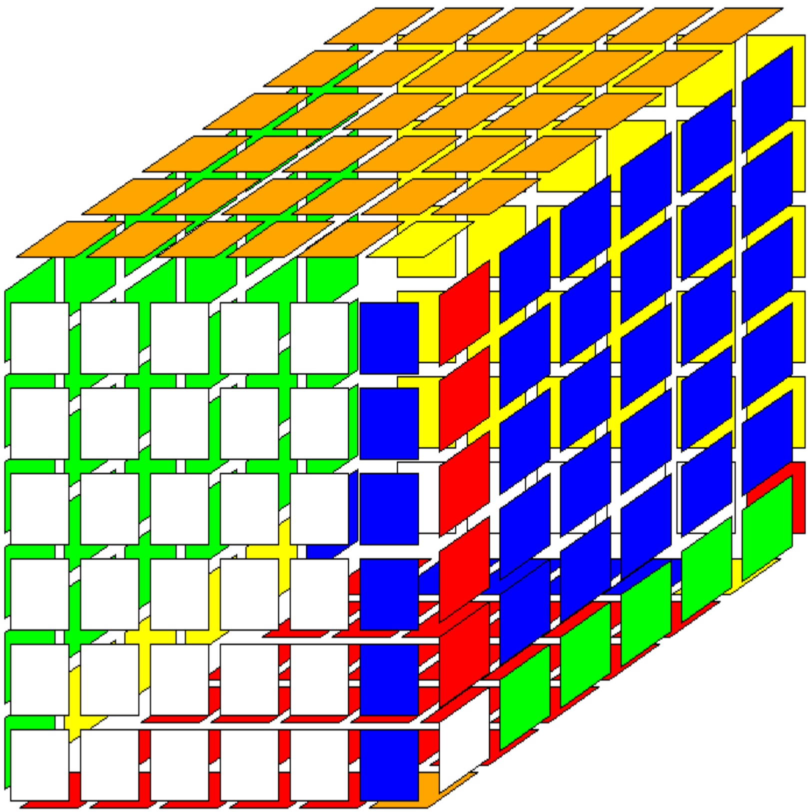
$W_1 = \mathrm{rot}(1,F)\mathrm{rot}(1,D)\mathrm{rot}^{-1}(1,F)\mathrm{rot}^{-1}(1,D)$

inverse

$$W^{-1}_1 = \mathrm{rot}(1,D)\mathrm{rot}(1,F)\mathrm{rot}^{-1}(1,D)\mathrm{rot}^{-1}(1,F)$$
$$= ((h(F),\sigma_{F,h(F)}(1,1))(d(F),\sigma_{F,d(F)}(1,1)))$$
$$o\ ((d(F),\sigma_{F,d(F)}(1,n))(b(F),\sigma_{F,b(F)}(1,n)))$$
$$o\ ((F,1,n)(F,n,n))\ o\ P$$

avec P une permutation qui laisse invariants les (U,i,j)

Cas 2



$$W_2 = W_1 rot(1,F) rot^{-1}(1,D) rot^{-1}(1,F) rot^{2}(1,D)$$

inverse

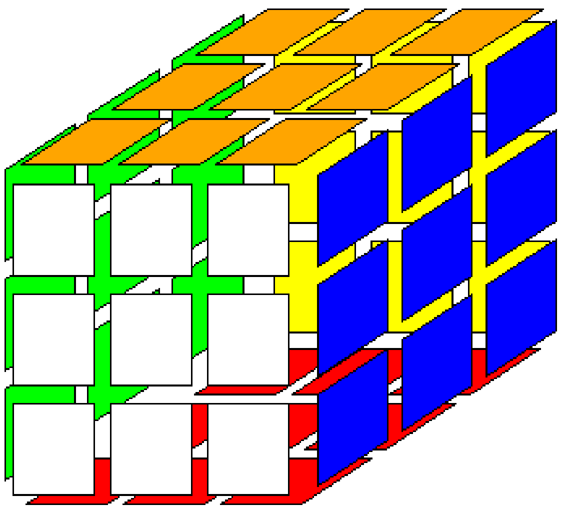
$$W_2^{-1} = rot^{2}(1,D) rot(1,F) rot(1,D) rot^{-1}(1,F) W_1^{-1}$$
$$= ((b(F),\sigma_{F,b(F)}(1,n))(h(F),\sigma_{F,h(F)}(1,1)))$$
$$o\ ((F,n,n)(d(F),\sigma_{F,d(F)}(1,n)))$$
$$o\ ((F,1,n)(d(F),\sigma_{F,d(F)}(1,1)))\ o\ P$$

avec P une permutation qui laisse invariants les (U,i,j)

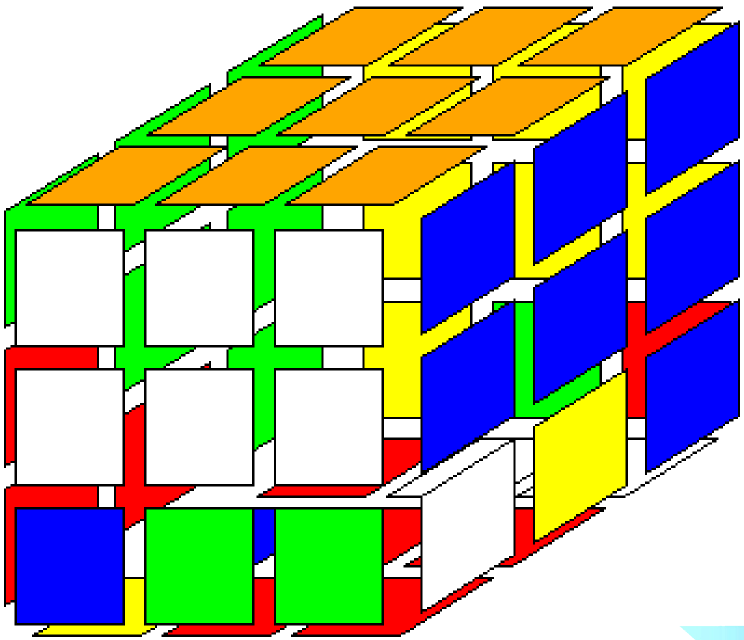
Utilisation

Soit X le nom de la première face. Il y a quatre coins que l'on numérote de 1 à 4. On applique ensuite exactement le même algorithme que précédemment en remplaçant les A_k par les W_k et en prenant i entre 1 et 4. Comme les W_k ne modifient pas les arrêtes de la première face, on se trouve dans un cas où le contour de la première face est fini.

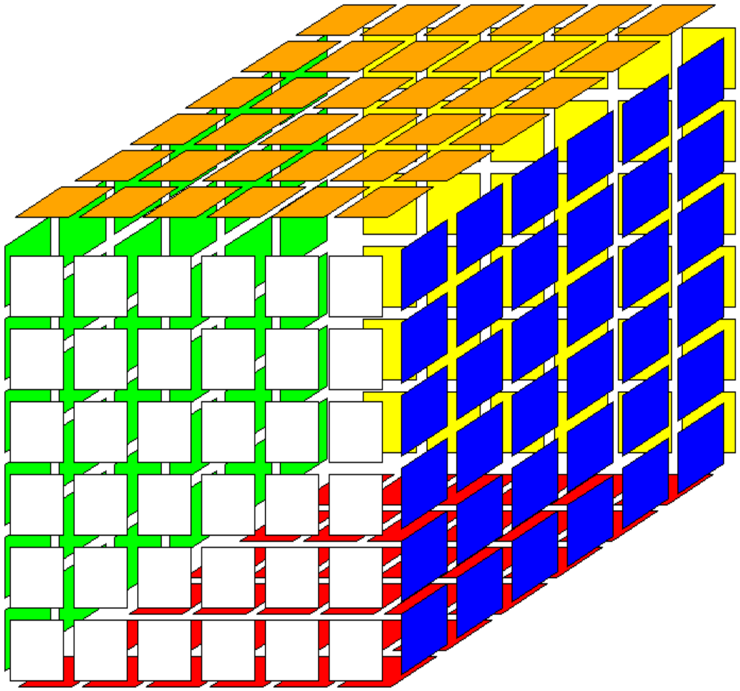
Contour des faces adjacentes (Première moitier)

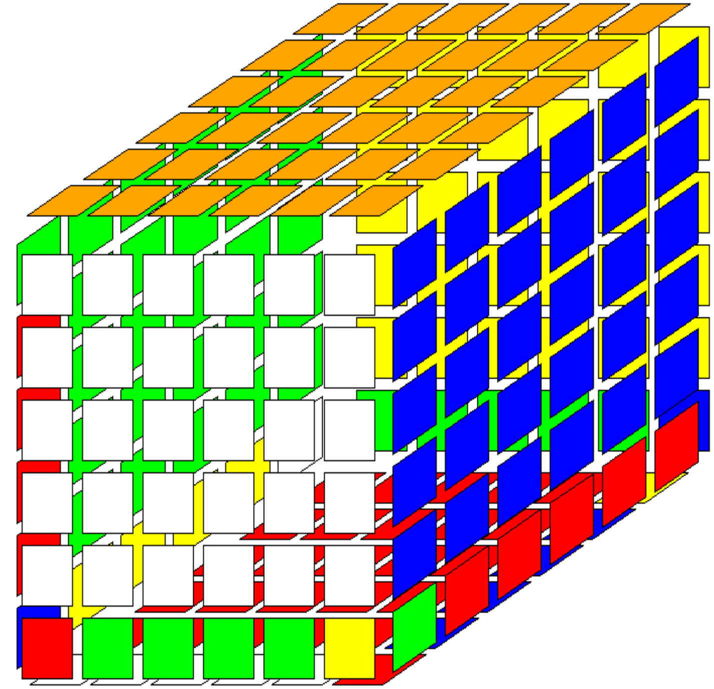


Sur un 3x3x3

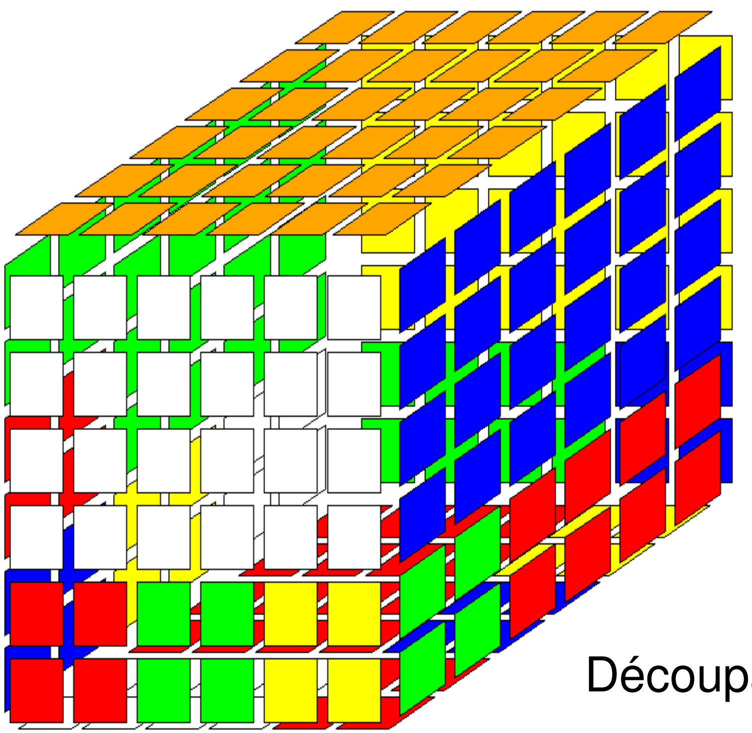


$Q_1 = rot(1,D)rot(1,L)rot(1,D)rot^{-1}(1,L)$
$rot^{-1}(1,D)rot^{-1}(1,F)rot^{-1}(1,D)rot(1,F)$





Découpage 1 + 4 + 1



Découpage en 2 + 2 + 2

$$Q_k = \mathrm{rot}_k(D)\mathrm{rot}_k(L)\mathrm{rot}_k(D)\mathrm{rot}_k^{-1}(L)\mathrm{rot}_k^{-1}(D)\,\mathrm{rot}_k^{-1}(F)\mathrm{rot}_k^{-1}(D)\mathrm{rot}_k(F)\mathrm{rot}_k^{2}(D)$$

$$Q_k^{-1} = ((d(F),\sigma_{F,d(F)}(1,n-k))(F,k,1)) \circ ((g(F),\sigma_{F,g(F)}(1,n))(F,n,n-k)) \circ P$$

avec P une permutation qui laisse invariants les (U,i,j) et les (X,i,j) pour X dans {F,L,B,R} et i dans [|1,k+1|]

Utilisation

Soit X le nom de la première face. Une moitier d'arrête comprend O = (N-2)/2 pièces quand N est pair et O = (N-1)/2 pièces quand N est impair, on les numérote de 1 à O. Il y a quatre arrêtes latérales que l'on numérote de 1 à 4. On applique ensuite l'algorithme suivant :

Pour i allant de 1 à O, on applique l'algorithme suivant :

Pour j allant de 1 à 4 on regarde si la i-ème pièce de l'arrête j est bien placée, si c'est le cas, on passe à j+1. Sinon :

- soit la pièce est à la place d'une autre pièce (i,k) (avec k > j) dans ce cas, on la retire en utilisant Q_i^{-1}(*), on fait tourner opposé(X) jusqu'à avoir la disposition décrite ci-avant puis enfin on la replace en utilisant un Q_i(*).

- soit la pièce est sur l'une des arrêtes mais pas à la place d'une autre pièce (i,k). Dans ce cas on applique rot_i(opposé(X)) jusqu'à ce qu'elle soit sous sa bonne position, on la retire en utilisant Q_i^{-1}(*) puis on termine comme le cas précédent.

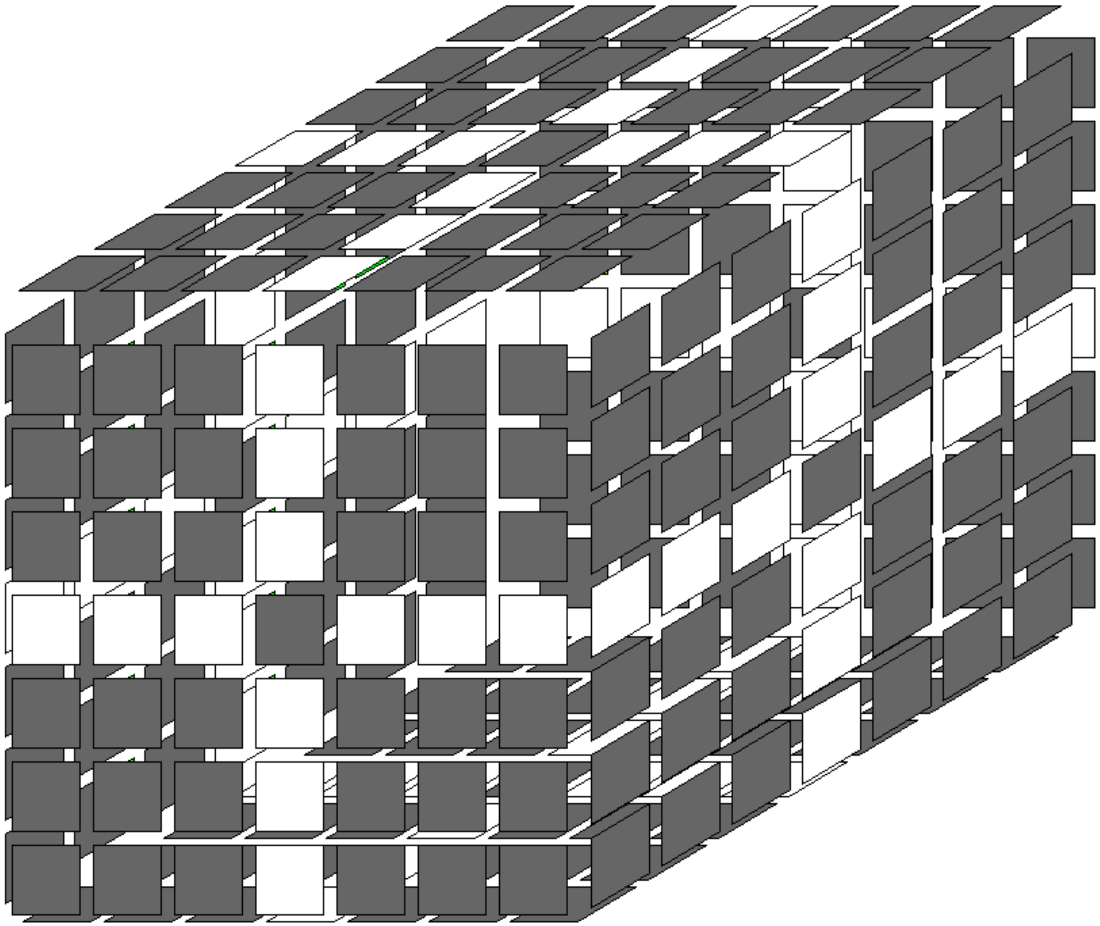
- soit on peut utiliser directement un Q_i^{-1} (*).

Puis on passe à i+1.

Comme les Q_k ne modifient pas les couronnes 1 à k, on a bien le résultat voulu.

Les coins de la dernière face (cas impair)

Il s'agit d'assimiller votre cube (2n+1)x(2n+1)x(2n+1) à une cube 3x3x3 en redécoupant celui-ci en (n+1+n)x(n+1+n)x(n+1+n).



Morphismes associés

Soit Δ : $C_3 \to C_{2n+1}$

$Id_{C3} \to Id_{C2n+1}$

$rot(X) \to rot_n(X)$

Il est immédiat que Δ définit un morphisme (injectif) entre C_3 et C_{2n+1}.

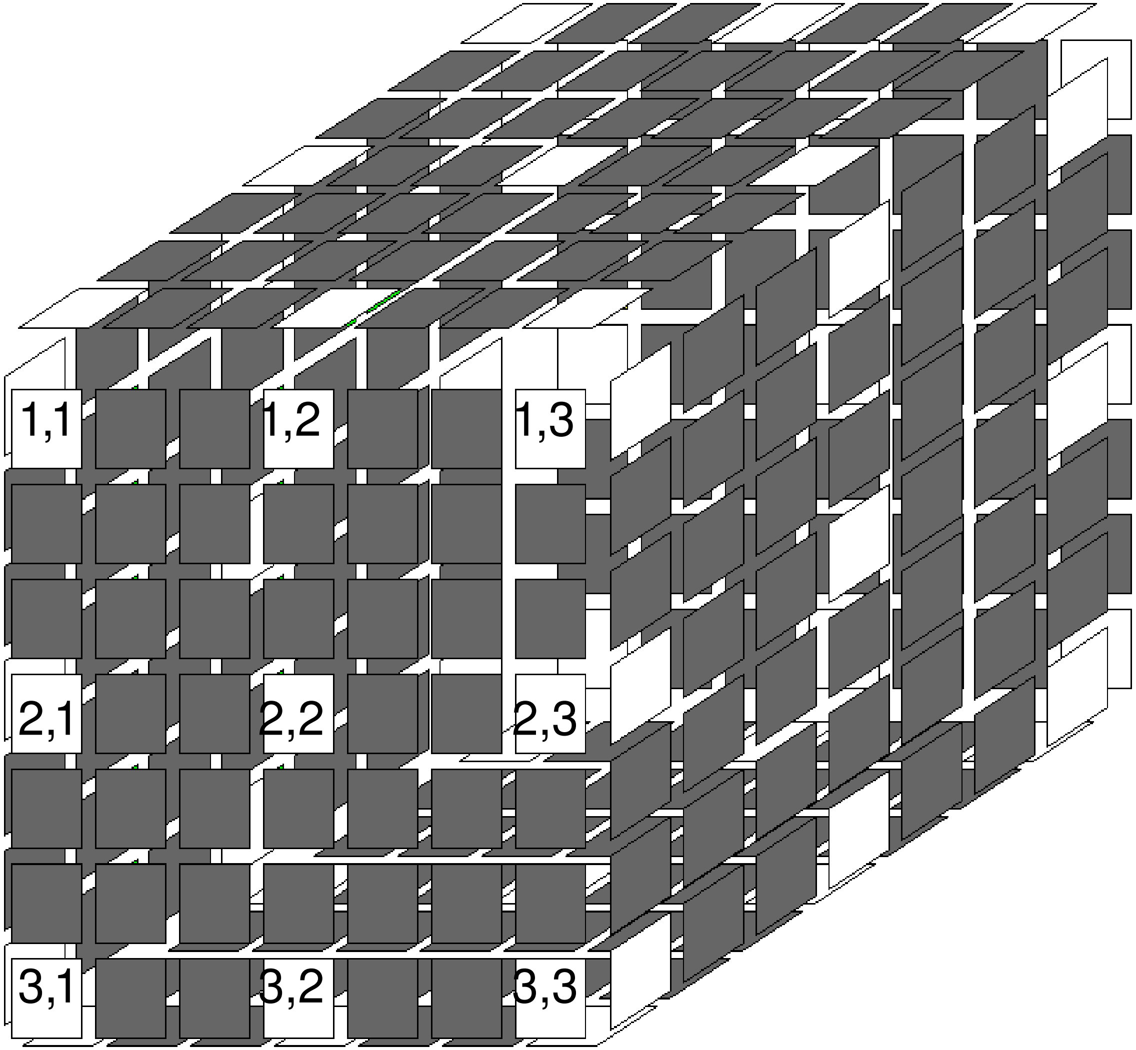
Soit ∇ : $C_{2n+1} \to C_3$

$Id_{C2n+1} \to Id_{C3}$

$$rot(i,X) \to \begin{cases} Id_{C3} \text{ si } i \neq 1 \text{ ou } n+1 \\ rot(1,X) \text{ si } i = 1 \\ rot(2,X) \text{ si } i = n+1 \end{cases}$$

∇ est aussi est un morphisme (surjectif), la démonstration est immédiate quand on voit que cela revient à renuméroter les coins, les centres d'arrête et les centres de face.

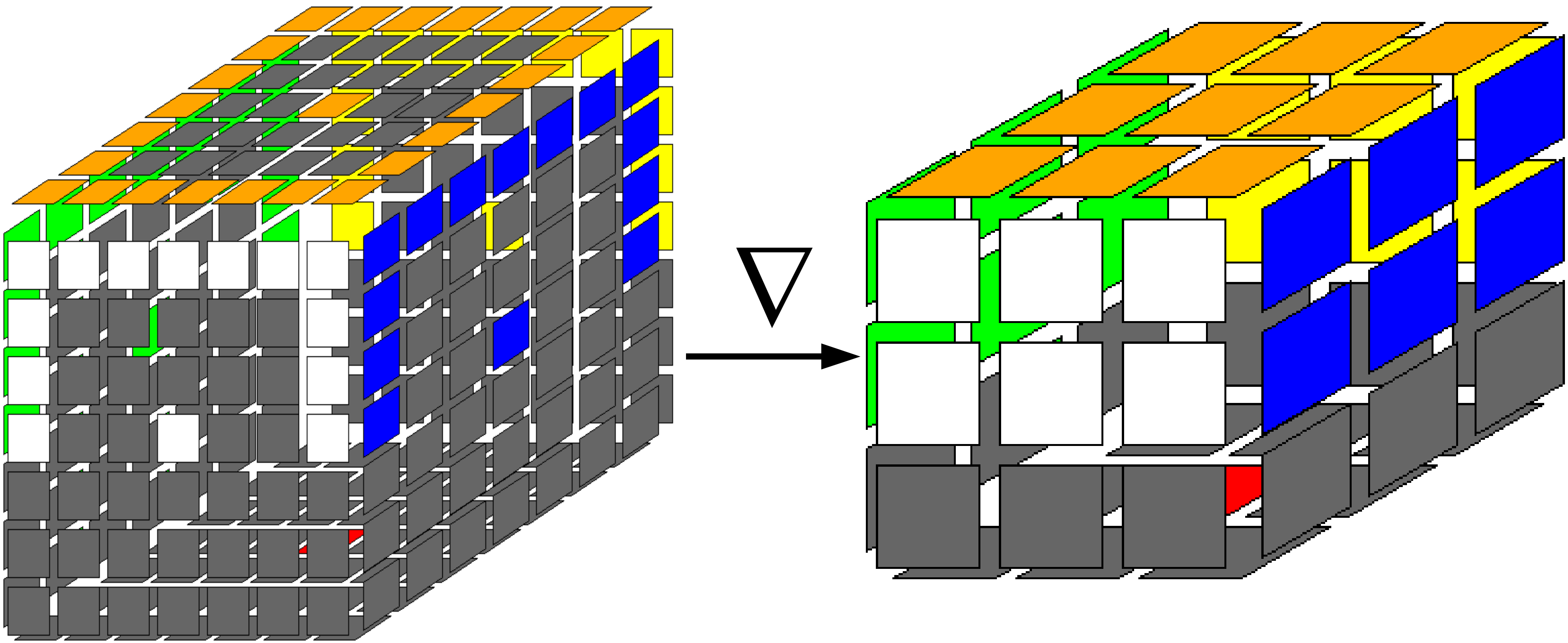
Morphisme ∇



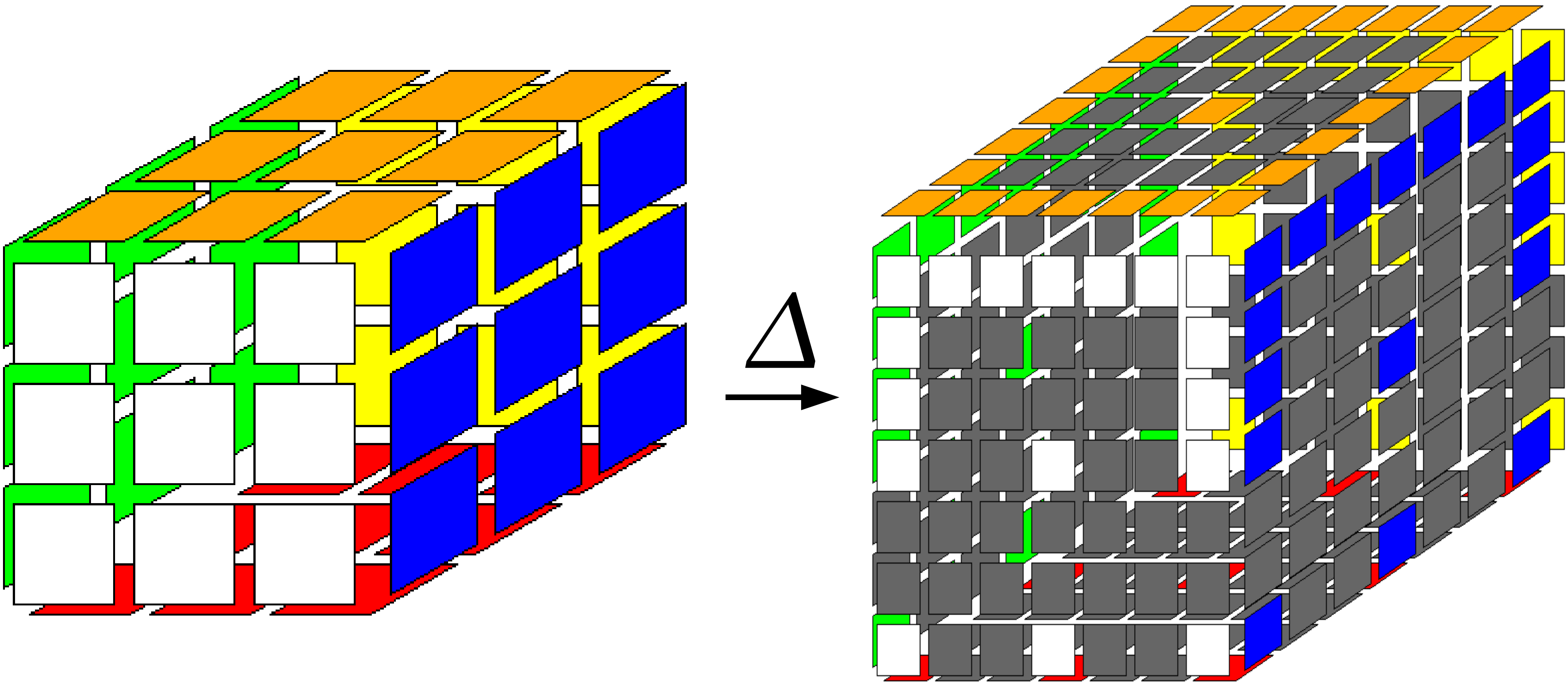
En effet, il sufit de voir ∇ comme un operateur qui concatène les pièces blanches de l'image ci-jointe.

Comment utiliser ∇ et Δ

Une fois que votre cube a ces contours faits
jusqu'à la moitier vous obtenez cette configuration :

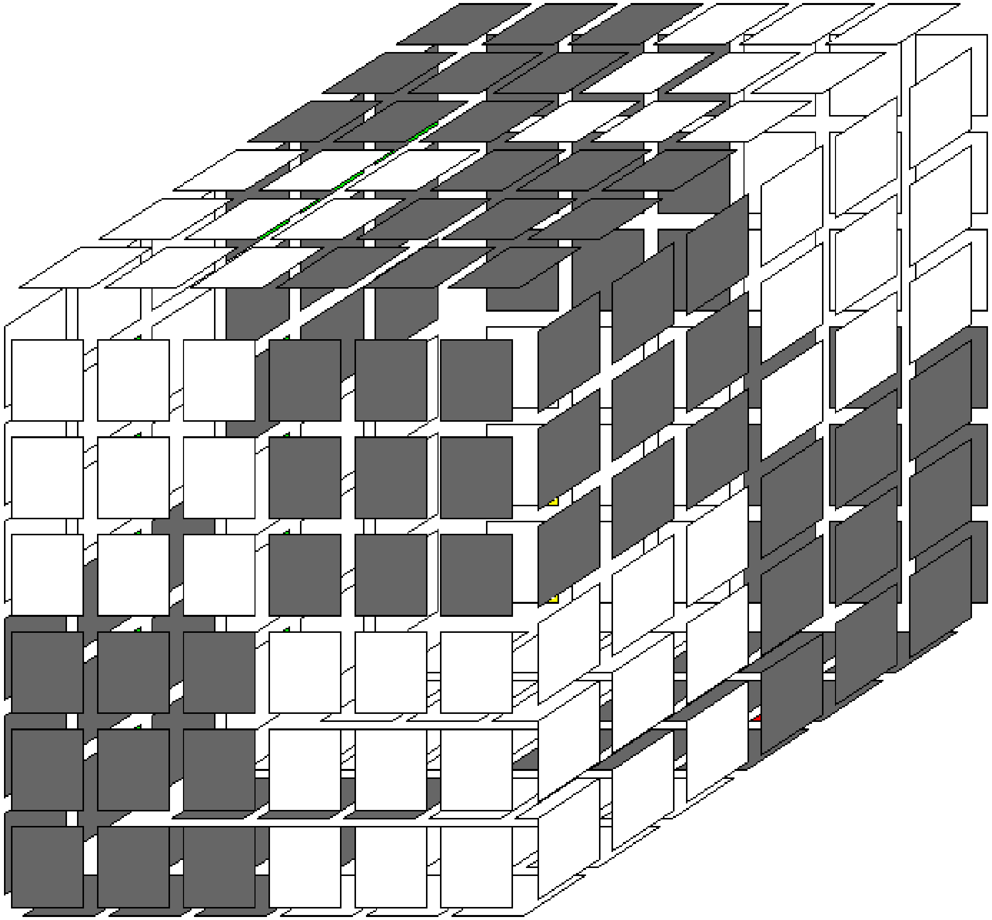


Puis vous résolvez votre cube en utilisant l'algorithme de résolution de la dernière face d'un 3x3x3



Les coins de la dernière face (cas pair)

Il s'agit de faire la même chose que pour le cas impair, sauf qu'au lieu de projeter sur un cube 3x3x3 on projette sur un 2x2x2.



Morphismes associés

Soit Δ : $C_2 \to C_{2n}$

$Id_{C2} \to Id_{C2n}$

$rot(X) \to rot_n(X)$

Il est immédiat que Δ définit un morphisme (injectif) entre C_2 et C_{2n}.

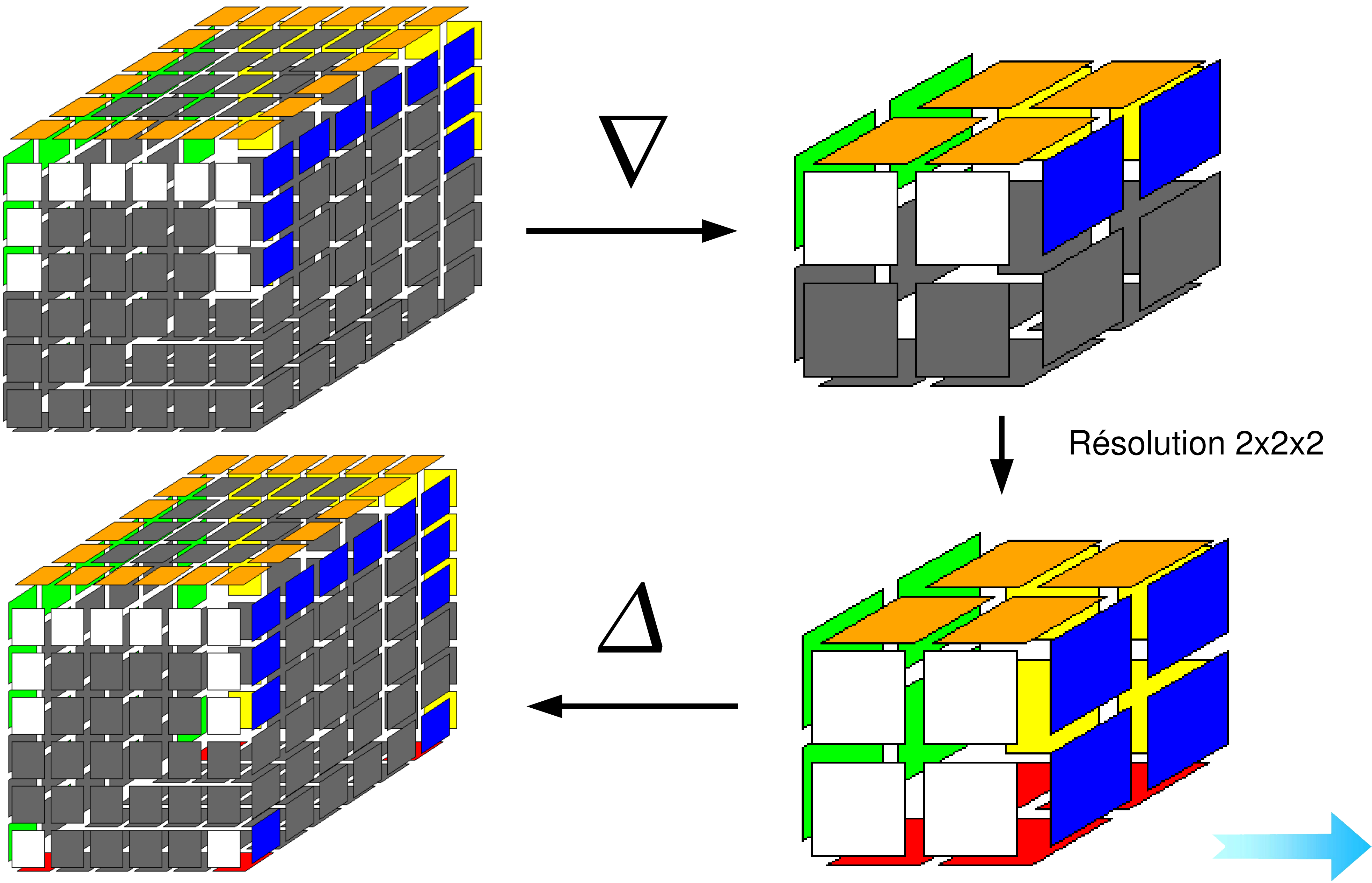
Soit ∇ : $C_{2n} \to C_2$

$Id_{C2n} \to Id_{C2}$

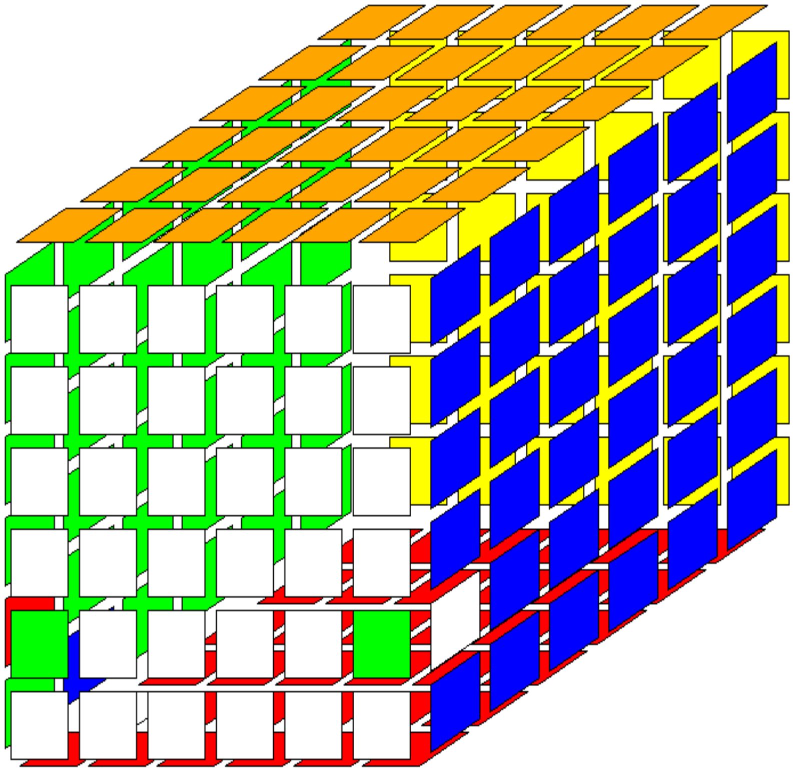
$$rot(i,X) \to \begin{cases} Id_{C3} & \text{si } i \neq 1 \\ rot(1,X) & \text{si } i = 1 \end{cases}$$

∇ est aussi est un morphisme (surjectif). L'explication étant la même qu'au cas impair.

Même schéma d'utilisation



Arrêtes de la dernière face



$J_i = rot^{-1}(1,F)rot(1,D)rot(1,F)rot^{-1}(i,D)$
$rot^{-1}(1,F)rot^{-1}(1,D)rot(1,F)rot(i,D)$

inverse

$J_i^{-1} = rot^{-1}(i,D)rot^{-1}(1,F)rot(1,D)rot(1,F)rot(i,D)$
$rot^{-1}(1,F)rot^{-1}(1,D)rot(1,F)$
$= [(g(F),\sigma_{F,g(F)}(1,n+1-i))(b(F),\sigma_{F,b(F)}(i,1))(F,n+1-i,n)]$
$o\ [(F,n+1-i,1)(g(F),\sigma_{F,g(F)}(1,n+1-i))(d(F),\sigma_{F,d(F)}(1,i))]$

Utilisation

On numérote les pièces les pièces des arrêtes de opposé(X) de 1 à 4(N-2). On applique ensuite l'algorithme suivant :

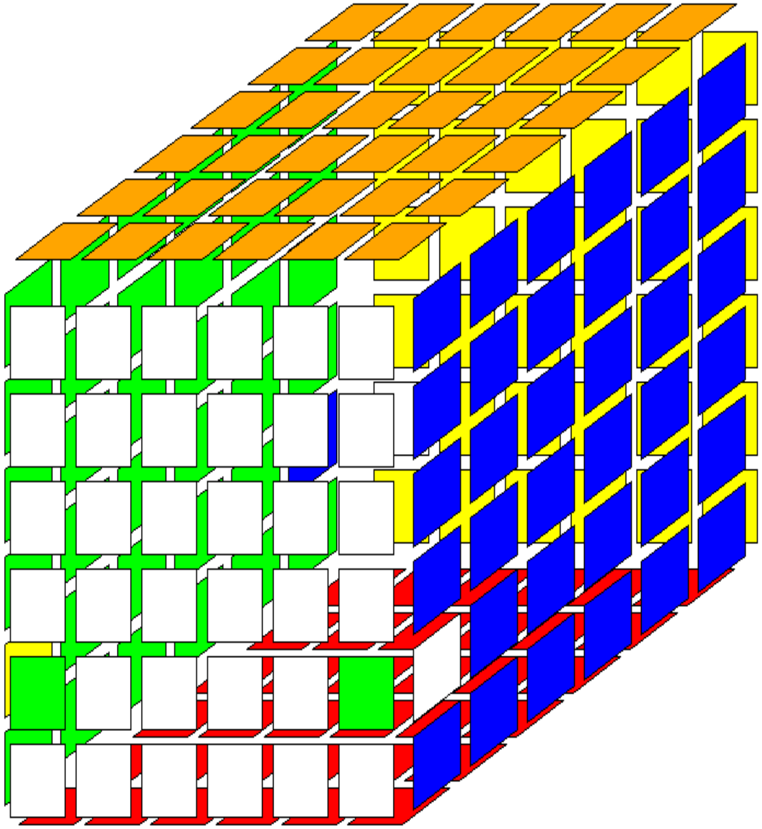
Pour i allant de 1 à 4(N-2) on regarde si la i-ème pièce de opposé(X) est bien placée, si c'est le cas, on passe à i+1.

Sinon :

- soit la pièce est à la place d'une autre pièce j (avec j > i) dans ce cas, on la retire en utilisant un des J_k (*), on applique rot(k,opposé(X)) jusqu'à ce que la pièce soit où il faut, puis on la replace en utilisant un J_k^{-1} (*).
- soit il suffit d'appliquer on peut utiliser rot(k,opposé(X)) jusqu'à ce que la pièce soit où il faut, puis on la replace en utilisant un J_k^{-1} (*).

Comme on ne déplace que des pièces qui ne sont pas au bon endroit, l'algorithme termine correctement quand i atteint 4(N-2).

Contour des faces adjacentes (Deuxième moitier)



$H_i = J_i rot(i,D) J_i rot^{-1}(i,D)$

inverse

$H_i^{-1} = rot(i,D) J_i^{-1} rot^{-1}(i,D) J_i$

= [(R,n+1-i,1)(F,n+1-i,n)(L,n+1-i,1)]

o [(F,n+1-i,n)(R,n+1-i,n)(B,n+1-i,n)]

l'ordre de H est 3

Utilisation

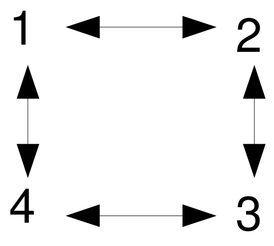
En utilisant le même argument qu'au début (cf :), on dénombre les cas possibles :

En appliquant rot(i,opposé(X)) un certain nombre de fois pour mettre une des pièce à sa place :

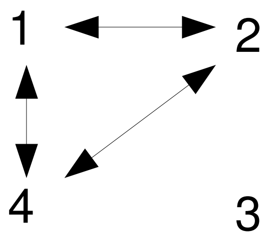
- soit les quatres pièces sont dans leurs bonnes positions.
- il ne peut pas y avoir que trois pièces bien mise sinon cela veut dire que la quatrième est retourné or c'est impossible.
- soit deux pièces sont bien placées et les deux autres inversées dans ce cas si vous appliquez encore des rotations et vous serez soit dans le cas suivant soit dans le même cas, alors il suffit d'appliquer un H_i est vous serez dans le cas suivant.
- soit votre pièce est la seule bien placée, dans ce cas il y a un H_i(*) qui résoud le problème.

Explication

Les morceaux d'arrête ayant une orientation fixe, le problème revient à créer l'identité dans Z = perm([|1,4|]) à partir des permutations qui font tourner les 4 chiffres dans un sens (rot) et des permutation qui font tourner 3 des 4 chiffres dans un sens (H).



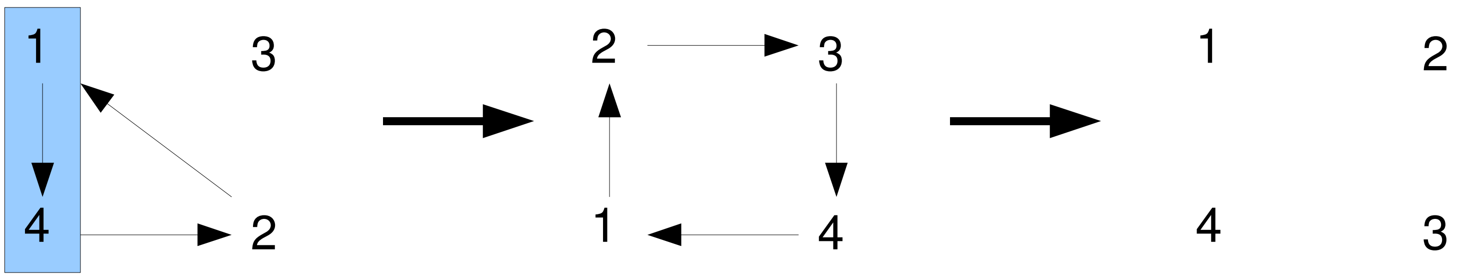
rot



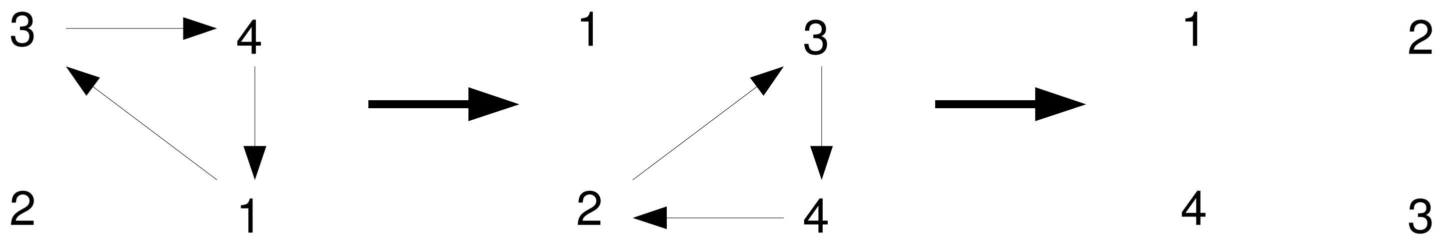
H

Les cas où deux pièces sont bien placées sont les suivants :

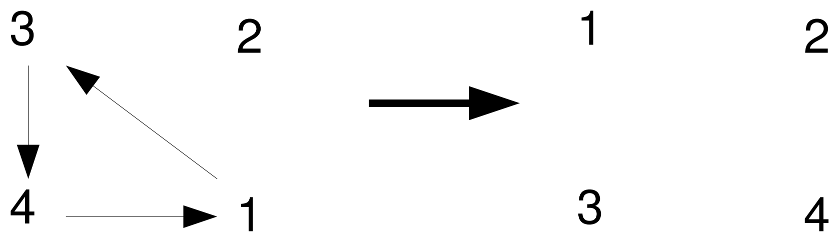
1)



2)



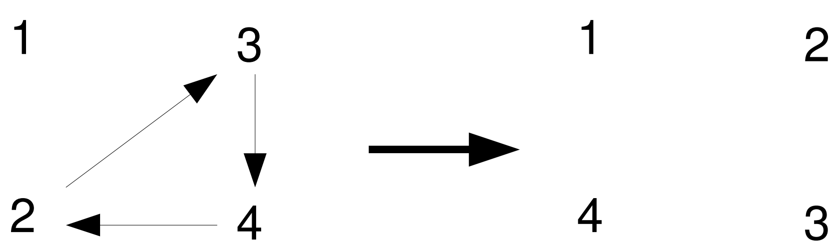
3)



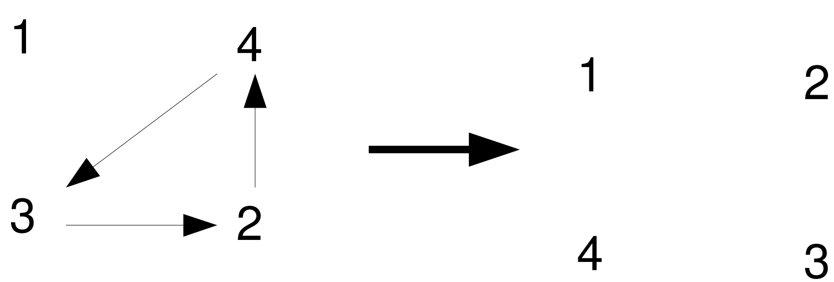
Cas impossible dans S_a donc aussi dans $S_{a,N,i}$!

Les cas où trois pièces commutent sont les suivants :

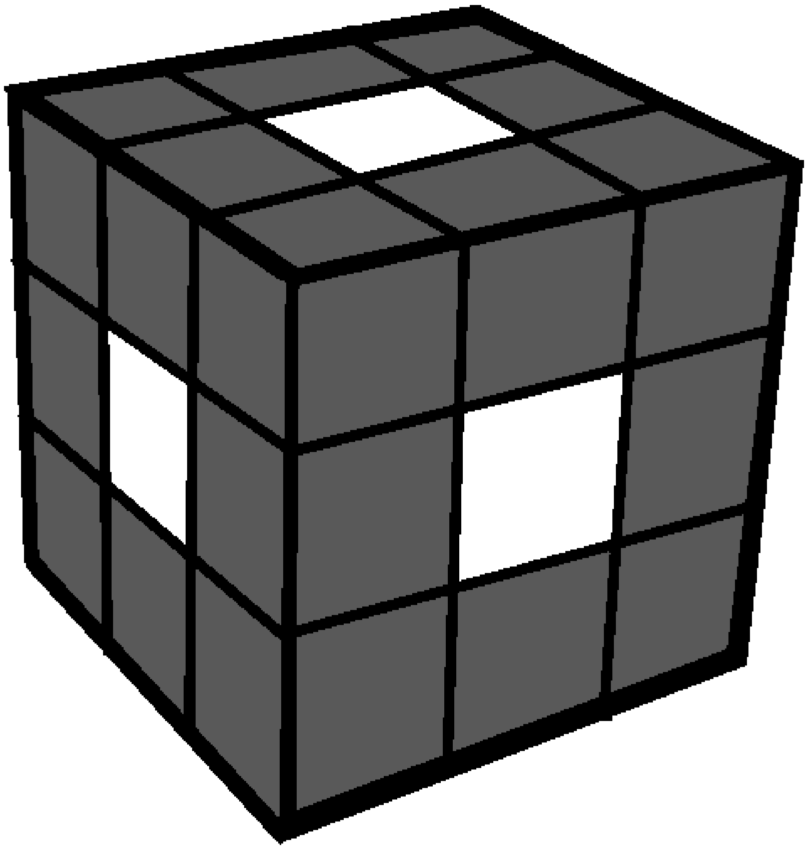
1)



2)



Fin des contours



L'intérieur des faces

C'est la partie la plus simple (il suffit de connaître deux types d'éléments de C_n) mais c'est de loin la plus longue de la résolution car plus le cube est gros, plus le temps à y passer est long (le rapport n'étant pas linéaire du tout).
On résoud chacune des faces en commencant par les couronnes du bord vers l'intérieur. Pour accélérer un peu les choses, mieux vaut il d'abord commencer par faire un trièdre puis son complémentaire afin de ne pas être obligé de défaire une partie de ce que l'on a déjà fait.

Les deux derniers éléments à connaître permettent de faire une permutation circulaire de segments et une permutation diagonale de carrés mais leur formules (qui dépendent de la profondeur de la couronne et des éléments que vous voulez bouger dans celle-ci) sont très difficilement explicitables, c'est pourquoi au lieu de donner ces dernières je vais plutôt essayer de les expliquer avec l'interface graphique du solveur.