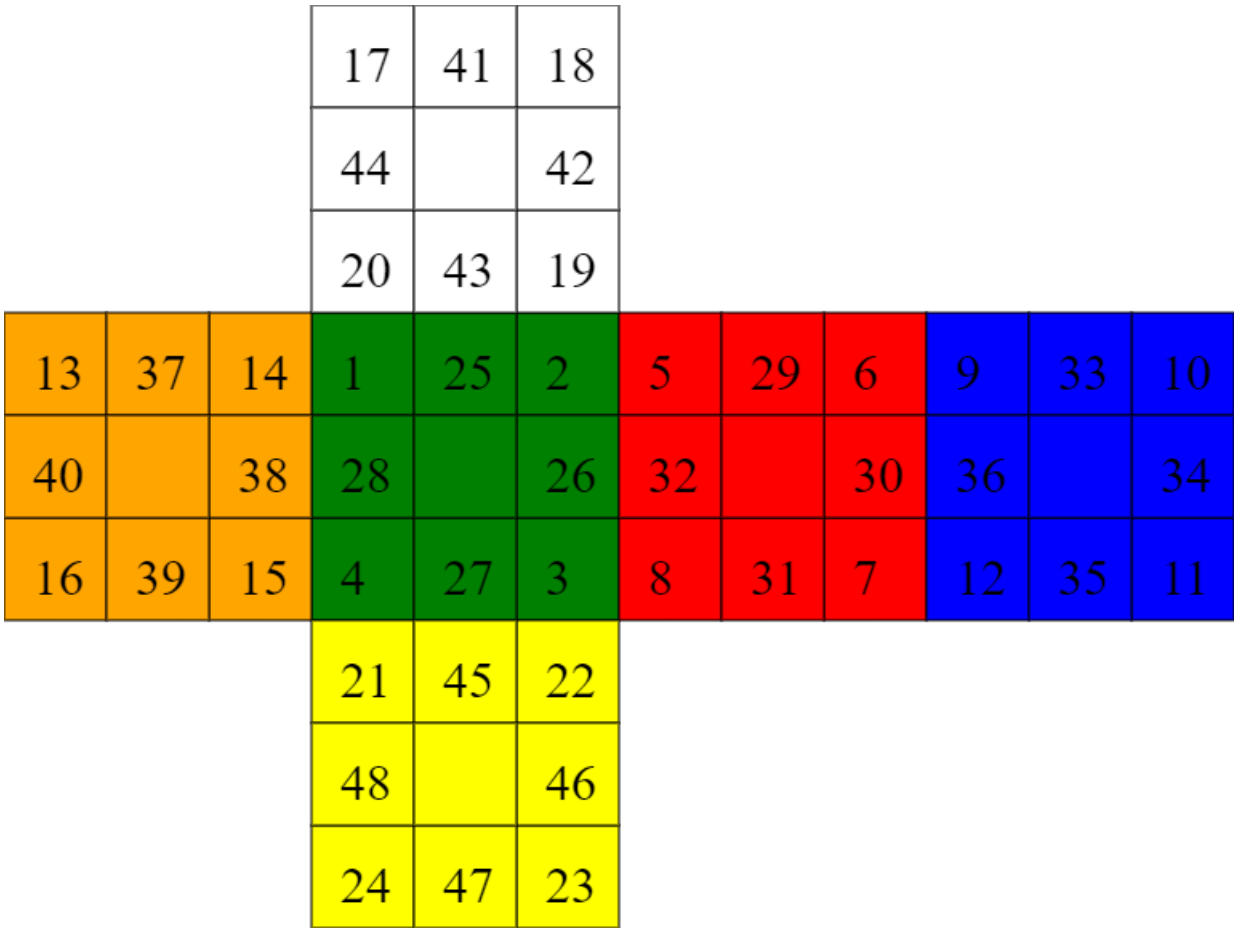
Page Rubik’s Cube de Cyclonium

Par Vincent Lesbros, 21 septembre 2024.

www.cyclonium.com/atelier/jeux/rubicks/index.html

Le bouton « *Reset* » affiche le cube dans sa position initiale, avec la numérotation des facettes adoptée, en 2D, sous forme de patron.



Il y a 48 facettes sur les 6 faces du cube.
Les facettes des coins ont été numérotées en premier, puis les arrêtes, en tournant dans le sens des aiguilles d’une montre, avec un ordre arbitraire pour les faces :
1 - face Avant, vert.
2 - face Droite, rouge.
3 - face Postérieure, bleue.
4 - face Gauche, orange.
5 - face Haut, blanche.
6 - face Bas, jaune.
Les facettes centrales ne sont pas numérotés car elles sont invariantes : tous les mouvements seront effectués en gardant cette orientation du cube.

On considère qu’un mouvement effectué sur le cube amène les facettes dans une nouvelle position, et que l’on peut noter cette position sous forme d’une permutation de ces 48 éléments.

La position initiale correspond à la permutation identité I :
1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48
Quand on clique sur *Reset*, la permutation courante P est réinitialisée à cette permutation.

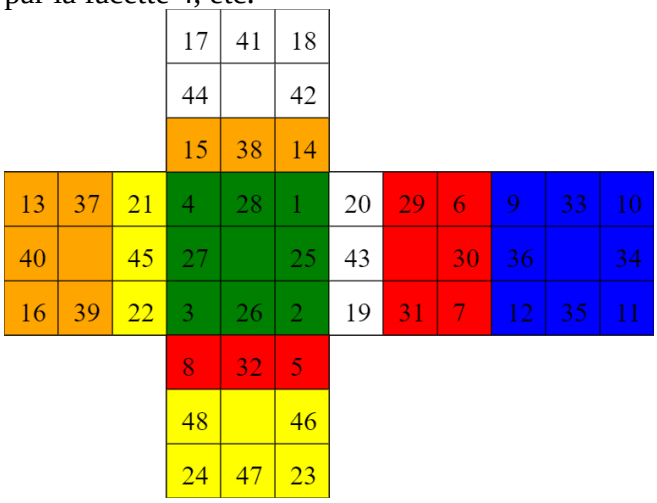
Les boutons A, D, G, H, B, P appliquent un mouvement d'un quart de tour dans le sens horaire à la face correspondante.

Par exemple, la permutation A, faisant tourner la face avant se note :
4-1-2-3-20-6-7-19-9-10-11-12-13-21-22-16-17-18-14-15-8-5-23-24-28-25-26-27-29-30-31-43-33-34-35-36-37-45-39-40-41-42-38-44-32-46-47-48

Vis-à-vis de la permutation I, cela donne :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
4	1	2	3	20	6	7	19	9	10	11	12	13	21	22	16	17	18	14	15	8	5	23	24	28	25	26	27	29	30	31	43	33	34	35	36	37	45	39	40	41	42	38	44	32	46	47	48

C'est-à-dire, qu'après le mouvement A, la position initiale de la facette 1 est maintenant occupée par la facette 4, etc.



Position des facettes après la rotation A appliquée sur le cube dans sa position initiale.

En interne, la pression du bouton A applique la composition de permutation :
P ⇐ A ° P.

Au fur et à mesure des pressions de touches A, D, G, H, B, P une chaîne se constitue.
Note : La chaîne se lit de gauche à droite, alors que la composition de permutation est effectuée de droite à gauche. Par exemple : la chaîne AABG correspond à la formule P ⇐ G°B°A°A°I.

On peut entrer une chaîne dans la zone de saisie de texte et utiliser le bouton « Exécuter » pour appliquer cette chaîne à la permutation courante.

Comme chaque mouvement de base est une rotation d'un quart de tour la répétition consécutive de 4 fois le même mouvement de base revient à la position précédente. On peut donc, dans une chaîne, éliminer tout mot de 4 lettres identiques consécutives sans changer la permutation atteinte par cette chaîne.
Exemple : la chaîne AAAAAA est équivalente à la chaîne AA.

Ma première version de la fonction « Simplifier » était la suivante :

```
function simplifierChaîne(chaîne) {

    let x = chaîne.replace(/AAAA/g, "");
    x = x.replace(/PPPP/g, "");
    x = x.replace(/DDDD/g, "");
    x = x.replace(/GGGG/g, "");
    x = x.replace(/HHHH/g, "");
    x = x.replace(/BBBB/g, "");

    return x;
}
```

Elle consiste en l'application successive d'expression régulières pour supprimer de la chaîne les séquences de 4 lettres identiques successives. Par la suite, une méthode plus générale a été mise en place.
Le bouton « Simplifier » effectue cette opération sur le texte de la zone de saisie.

D'autre part, on peut constater que chaque chaîne possède une chaîne inverse : il suffit de refaire les mouvements en sens contraire et dans l'ordre inverse pour restituer la position initiale du cube.
Je n'ai noté que les mouvement horaires. Donc pour inverser un mouvement de base, il faut le répéter 3 fois.
L'inverse de A est AAA. L'inverse de AB est BBBAAA.

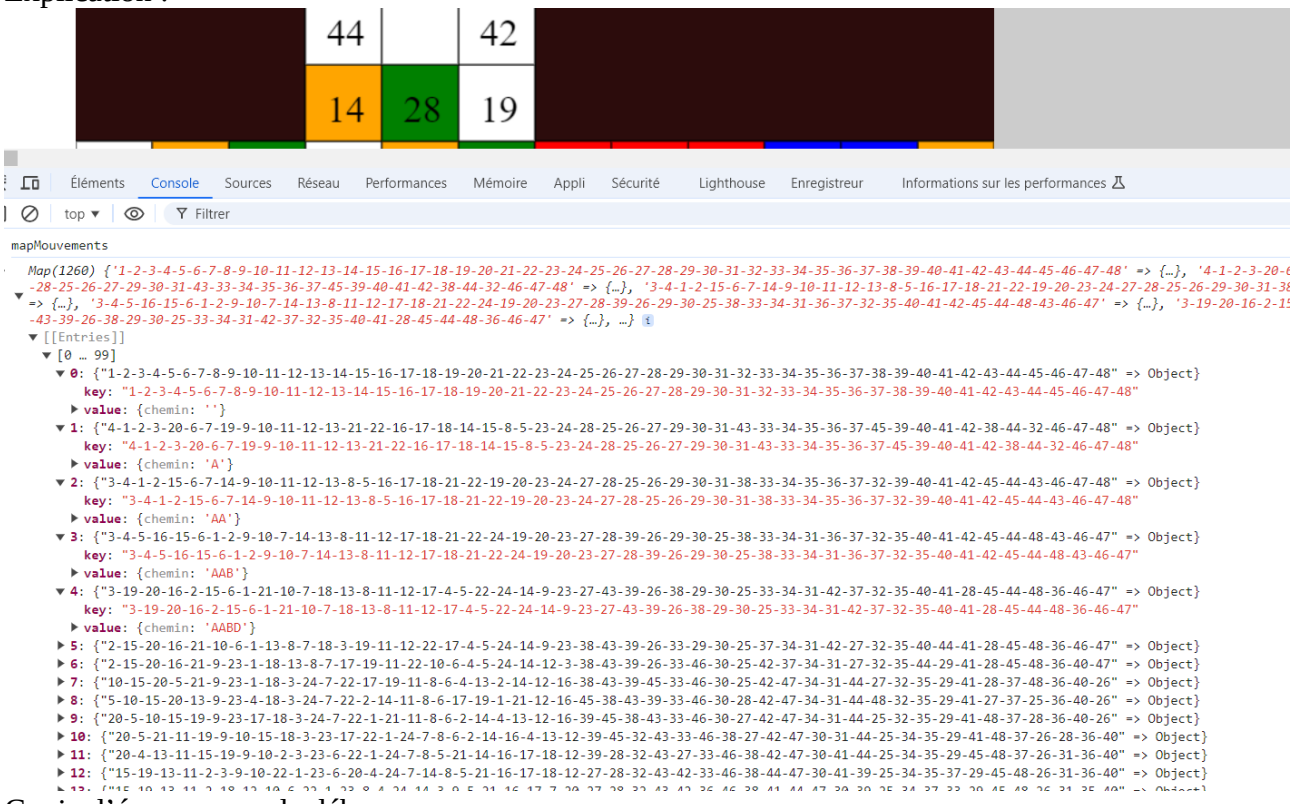
Le bouton « Inverser » effectue cette opération sur le texte de la zone de saisie.

L'ordre d'une chaîne est le nombre de fois qu'il faut l'appliquer pour revenir à l'identité.

Par exemple, l'ordre de I est 1. L'ordre de A est 4, l'ordre de AA est 2, l'ordre de AADD est 6, etc.

Le bouton « Ordre » calcule l'ordre de la chaîne saisie. Attention ! La mémoire est limitée et il existe des chaînes dont l'ordre est très élevé.
Par exemple, l'ordre de AABDHPG est de 180, cela reste calculable, mais il y a des cas où mon algorithme demandera trop de mémoire pour être exécuter.

Explication :



Copie d'écran en mode débogage.

Je crée un dictionnaire, un objet ***Map*** en *Javascript*, avec pour clés les permutations atteintes et pour valeur le chemin rencontré le plus court pour y parvenir.
La variable le contenant est nommée `mapMouvements`.

Voici la fonction qui permet d’enregistrer les permutations dans le dictionnaire :

```
function ajouterMouvement(chaîne) {
    // on part de l'identité
    let p = mI; // permutation courante
    let cp = perm_clef(p); // clé de la permutation courante
    let chemin = ""; // chemin suivit dans la chaîne

    for (let k = 0; k <= chaîne.length; ++k) {
        let nœud = mapMouvements.get(cp);

        if (!nœud) {
            // nouveau nœud à créer
            nœud = {
                // permutation: p,
                chemin: chemin,
            }
            mapMouvements.set(cp, nœud);
        }

        if (nœud.chemin.length < chemin.length) {
            chemin = nœud.chemin;
        }
        else if (nœud.chemin.length > chemin.length) {
            // on a trouvé plus court !
            nœud.chemin = chemin;
        }

        // avancer d'une lettre
        if (k < chaîne.length) {
            let mouvement = chaîne[k]; // une lettre de l'alphabet
            switch (mouvement) {
                case "A": pM = mA; break;
                case "P": pM = mP; break;
                case "D": pM = mD; break;
                case "G": pM = mG; break;
                case "H": pM = mH; break;
                case "B": pM = mB; break;
                default:
                    continue;
            }
            chemin += mouvement;
            p = perm_produit(pM, p);
            cp = perm_clef(p);
        }
    }
    return chemin;
}
```

Le bouton « Simplifier » effectue la simplification en retournant le chemin ramené par cette fonction, donc, la chaîne la plus courte connue aboutissant à la même permutation que celle de la chaîne donnée.

Notes

1/ Je me suis contenté de 6 mouvements : APDGHB. On pourrait raccourcir les chaînes en admettant les chemins inverses : apdghb par exemple. En posant a = AAA, b = BBB, etc.

2/ Les facettes des coins, par exemple celles du coin vert-jaune-orange, 4-21-15, ne peuvent pas être dissociées. De même celles des arrêtes. Mais les coins peuvent se trouver dans trois orientations différentes et les arrêtes dans deux différentes.
J’ai commencé par vouloir dissocier la description des mouvements des coins et celle des arrêtes, mais le plus simple a été de considérer S_{48} comme étant le support de base, au lieu de S_8 pour les coins plus les permutations correspondant aux trois orientation, et S_{12} pour les arrêtes avec la prise en compte des deux orientations.
En pensant à la représentation des permutations par croisements de fils, je me demande s’il n’y a pas une nouvelle sorte de permutation à considérer : où les fils ne représenterait pas des éléments uniques, mais seraient des « torons », contenant plusieurs fils, et pourraient arriver à leur place avec un ordre associé (une permutation circulaire en l’occurrence).

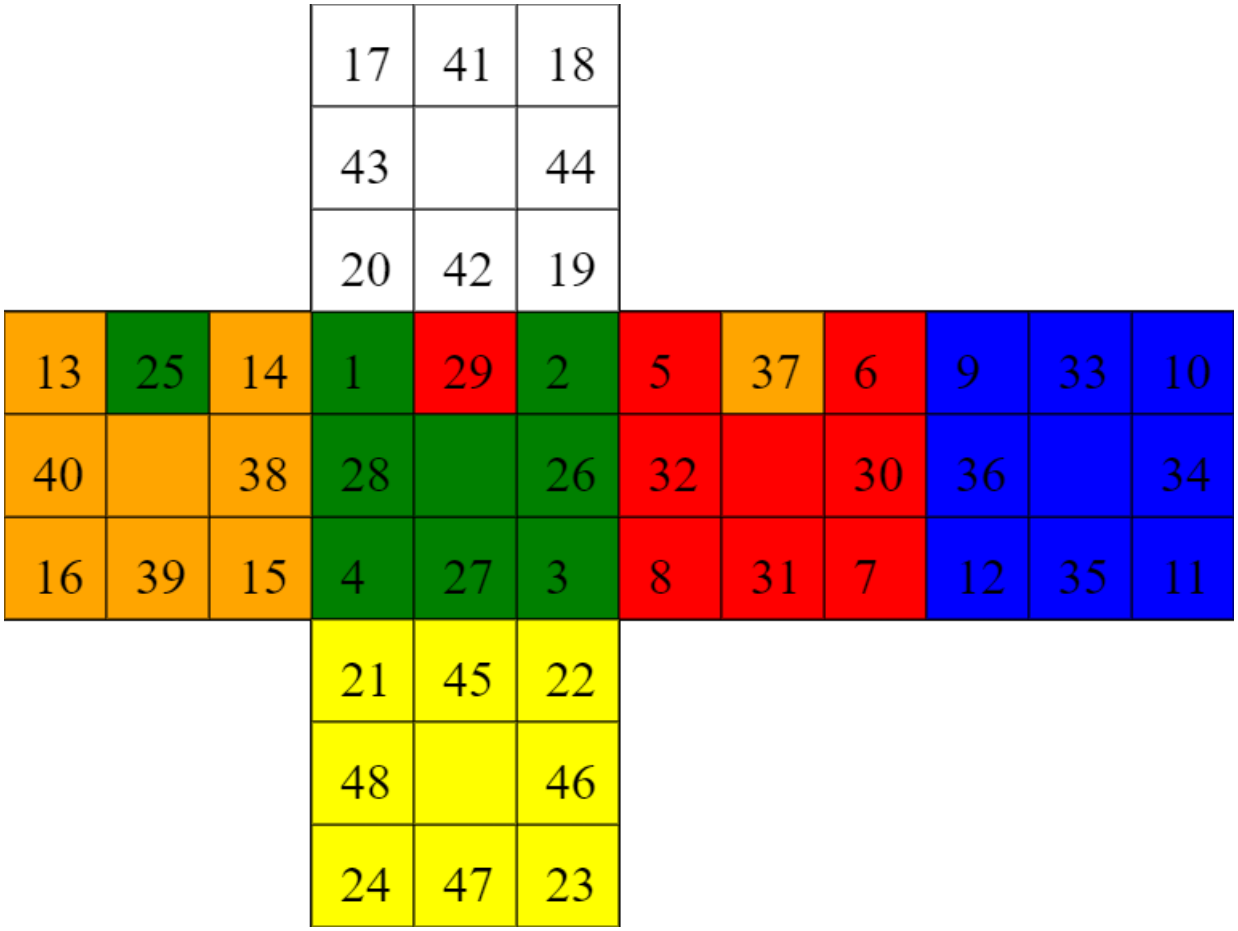
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
4 1 2 3 20 6 7 19 9 10 11 12 13 21 22 16 17 18 14 15 8 5 23 24 28 25 26 27 29 30 31 43 33 34 35 36 37 45 39 40 41 42 38 44 32 46 47 48
3 4 1 2 15 6 7 14 9 10 11 12 13 8 5 16 17 18 21 22 19 20 23 24 27 28 25 26 29 30 31 38 33 34 35 36 37 32 39 40 41 42 45 44 43 46 47 48
2 3 4 1 22 6 7 21 9 10 11 12 13 19 20 16 17 18 8 5 14 15 23 24 26 27 28 25 29 30 31 45 33 34 35 36 37 43 39 40 41 42 32 44 38 46 47 48
Permutations I,A, AA, AAA.

3/ La page actuelle permet de passer d’une chaîne à une permutation, pas l’inverse. Donc, elle ne permet pas de résoudre un cube connaissant sa position si on n’est pas arrivé à cette position par une chaîne connue.

Cela me fait penser que le cube crée un espace. Un espace où chaque point est une permutation ou une configuration. On peut circuler d’un point à l’autre en choisissant une des 6 lettres de notre alphabet APDGHB. Avancer 4 fois dans la même direction fait revenir au même point.
Une distance dans cet espace pourrait être la longueur de la chaîne minimale permettant de passer d’un point à l’autre.

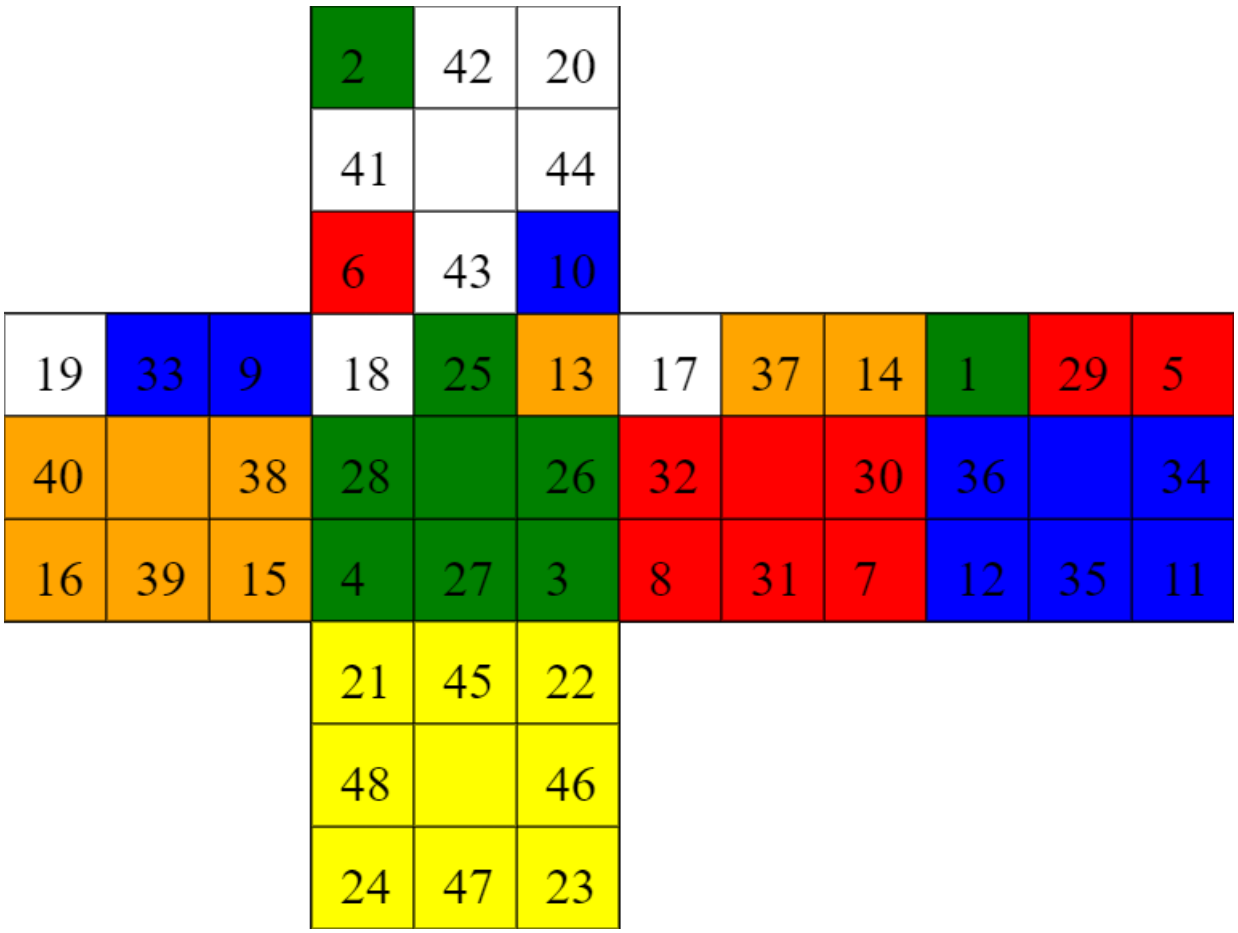
Les stratégies actuelles, pour résoudre le cube, sont basées sur des formules (des chaînes) permettant de permuter certains éléments sans changer la position des autres.

Par exemple, la chaîne AAHGDDDAAGGGDHAA effectue une permutation circulaire de trois arrêtes, sans modifier le reste du cube.



Chaîne : AAHGDDDAAGGGDHAA d’ordre 3.

Pour en permuter trois autres, on peut préfixer la chaîne par un mouvement, GP, par exemple, et suffixer par son inverse. Cela donne : GPAAHGDDDAAGGGDHAAPPPGGG



Autre exemple, la chaîne DHDDDHDHHDDD, d'ordre 6, ne modifie que la couronne du haut du cube et permet de manipuler les coins du haut sans toucher aux deux étages du bas déjà construits.

La résolution du cube est donc la recherche de la chaîne (la plus courte si possible), permettant d'obtenir la permutation souhaitée.