UNE ANALYSE DU RUBIK'S CUBE

INTRODUCTION :

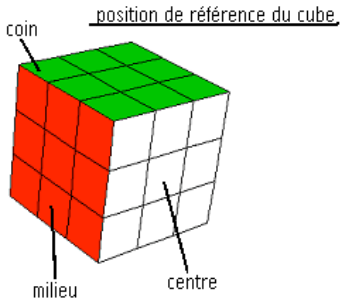
LE RUBIK'S CUBE standard (3x3x3), objet multicolore avec lequel tout enfant a un jour joué (en en décollant les étiquettes tout du moins), se présente sous la forme d'un petit cube large de quelques centimètres découpé en 26 pièces mobiles et au centre duquel se loge l'ingénieux mécanisme de rotation. Sa résolution n'est pas moins intrigante que son fonctionnement interne pour le néophyte.

Erno Rubik, architecte hongrois, eut l'idée du Rubik's cube en 1974. Il fait partie de la famille des casse-têtes « permutations ». A Toronto, au Canada, en Aout 2003, s'est déroulé le Championnat mondial officiel du Rubik's Cube. Des mathématiciens et informaticiens ont effectué des recherches très poussées sur le sujet ; de nombreux ouvrages lui ont été consacrés. L'objectif de mon TIPE n'a cependant pas été de faire un résumé des connaissances sur le sujet, mais plutôt de créer ma propre vision de cet objet et de ses liens avec les mathématiques et l'informatique. La bibliographie est donc très peu étoffée, et les conventions que j'ai adoptées ne sont pas toujours celles utilisées ailleurs. Ce qui m'a attiré dans ce sujet est sa capacité à concrétiser des mathématiques abstraites.

I) MISE EN PLACE D'UNE STRUCTURE DE GROUPE SUR LE CUBE.

- Dénombrement des positions :

Les centres des faces sont fixés et ne peuvent pas être déplacés. Le squelette du cube est donc constitué par le mécanisme de rotation (logé dans le petit cube central) et des 6 centres des faces dont les positions relatives sont fixées. **On observe le cube sous un angle fixé arbitraire appelé position de référence** (et à partir de cette position, on dénomme les 6 faces: face du haut, face du bas, droite, gauche, avant, arrière.)



On peut également dénommer les faces par la couleur de leur milieu (V pour vert, W : blanc, R : rouge, B : bleue, O : orange, J : jaune).

On note G l'ensemble des positions du cube. Card(G)= 519024039293878272000 ~ 5.2e20

- Encodage d'une position :

On voudrait bien pouvoir associer à chaque position du cube un objet mathématique la représentant. On numérote arbitrairement les coins de 1 à 8 et les milieux de 1 à 12. **On code une position quelconque du cube en la représentant par l'ensemble des transformations qui permettent de passer du cube fait (l'« identité ») à la position que l'on souhaite coder**. Pour déterminer l'encodage d'une position, on place le cube en position de référence, et l'on code successivement les déplacements des coins C1..C8, puis des milieux M1.. M12. Chaque coin prend la place d'un autre (éventuellement de lui-même) et deux coins ne peuvent se trouver au même endroit en même temps. Il en va de même pour mes milieux. **Il s'agit donc d'une permutation des coins Sigma-C et d'une permutation des milieux Sigma-M.** A partir de Sigma-C, on construit une **matrice de permutations des coins PC** de dimension 1*8 ; PC = [Sigma-C(C1) … Sigma-C(C8)]. On définit de même une **matrice de permutations des milieux PM**.

Cependant, la donnée de ces deux permutations ne caractérise pas entièrement une position du cube. En effet, il faut également prendre en compte un autre paramètre : la rotation possible des coins et des milieux après leur déplacement. Pour ce faire, **il a fallu établir un ensemble de déplacements de référence des coins et des milieux, à partir desquels ont peut associer à chaque milieu et coin un indice de rotation qui quantifie l'écart à une orientation de référence**. Toute la difficulté de cette recherche résidait dans le fait qu'il fallait que ces indices de rotation aient un sens du point de vue du Rubik's Cube, et surtout, comme on le verra, que l'on puisse les composer par la loi qui fera du Rubik's Cube un groupe. L'indice de rotation d'un coin Ci (1<=i<=8), noté ICi peut prendre 3 valeurs : 0 (le coin est bien orienté), -1(il a fallut tourner le coin de 120° dans le sens horaire pour que son orientation corresponde à la position que l'on souhaite coder) ou 1 (il a fallut tourner le coin de 120° dans le sens trigonométrique pour que son orientation corresponde à la position que l'on souhaite coder). L'indice de rotation d'un milieu Mi(1<=i<=12), appelé IMi, peut prendre 2 valeurs : 0 (le milieu est bien orienté), ou 1 (il a fallut faire pivoter le milieu d'un demi-tour pour que son orientation corresponde à la position que l'on souhaite coder). On représente ces indices de rotations sous la forme de matrices :
matrice des indices de rotation des coins (IRC) : [IC1 IC2 ... IC8]
matrice des indices de rotation des milieux (IRM) : [IM1 IM2 … IM12]

Une position du cube est entièrement déterminée par la donnée de ces 4 matrices PC, PM, IRC, IRM.
On définit l'encodage d'une position comme une 4-liste de la forme : { PC, PM, IRC, IRM }

Une position 'p' du cube sera donc représentée par une 4-liste constituée par les 4 matrices caractéristiques de cette position. Cette 4-liste est appelé transcription de p et est notée T(p) .
Par la suite, on confondra une position et sa transcription.

Transcription d'un mouvement : c'est la transcription de la position du cube obtenue en effectuant ce mouvement sur le cube fait.

Un exemple concret : pour coder le mouvement « demi-tour de la face du bas », on prend le Rubik's Cube fait, et on effectue un demi-tour de la face du bas. On détermine alors les 4 matrices caractéristiques de la position obtenue. La 4-liste que l'on obtient ainsi est appelée « transcription du mouvement 'demi-tour de la face du bas'. » Dorénavant, la rotation d'un quart de tour dans le sens trigonométrique d'une face sera représentée par la lettre majuscule caractéristique du nom de la face : H pour haut, P pour bas (=postérieure), D pour droite, G pour gauche, A pour arrière, F pour face ; la lettre minuscule désignera un quart de tour d'une face élémentaire dans le sens horaire. Par la suite, on confondra un mouvement et la position qu'il engendre sur le cube-identité.

- Définition de la loi de produit semi-direct * ; structure de groupe du Cube (G,*):

On note encore G l'ensemble des transcriptions des positions du cube, càd que G =
$\{ \{\sigma\text{-c}, \sigma\text{-m}, [IC1, IC2, …., IC8], [IM1, IM2, …, IM12]\} / \sigma\text{-c} \in S8, \sigma\text{-m} \in S12, (IC1,…,IC8) \in \{-1 ;0 ;1\}^8$ et $(IM1,…,IM12) \in \{0 ;1\}^{12}\}$

On définit alors la loi * de produit sur G par X*Y=
$\{(\sigma c')o\ (\sigma c),$
$(\sigma m')o(\sigma m),$
$[IC_1 + IC'_{\sigma c(1)}[3]\ ,\ IC_2 + IC'_{\sigma c(2)}\ [3],\ldots\ IC_8 + IC'_{\sigma c(8)}\ [3]],$
$[IM_1 + IM'_{\sigma m(1)}\ [2],\ IM_2 + IM'_{\sigma m(2)}\ [2],\ldots\ IM_{12} + IM'_{\sigma m(12)}\ [2]]\}$

On montre que (G,*) est un groupe non abélien appelé groupe du Cube

En fait cette loi de composition retranscrit le fonctionnement réel du Rubik's Cube.
Ainsi, si l'on compose les transcriptions de deux mouvements élémentaires, on obtiendra la transcription de la composée de ces deux mouvements élémentaires, c'est à dire de la position obtenue après avoir effectué à partir du cube fait ces deux mouvements élémentaires. Formellement, on a pour deux mouvements X et Y:

T(YoX)=T(X)*T(Y) où o désigne la composition de ces deux mouvements sur le cube.

(Appliquer les mouvements X puis Y sur le cube s'écrit mathématiquement YoX.)

Cela peut paraître anodin mais c'est fondamental, car cela permet d'informatiser les recherches ; en effet, si l'on détermine (« à la main ») les transcriptions des 6 mouvements élémentaires (dans le sens trigonométrique), on pourra déterminer informatiquement la transcription d'une position quelconque en composant entre eux ces mouvements élémentaires.

- Invariants du Rubik's Cube :

Les seules mouvements que l'on est capable d'effectuer sur un Rubik's Cube sans le démonter sont les mouvements élémentaires et leurs composés, c'est à dire que toute manipulation que l'on effectue sur le cube peut se décomposer en une succession de mouvements élémentaires (c-à-d de quarts de tour d'une des 6 faces). De fait, certaines positions du cube ne seront pas accessibles à partir d'un Rubik's Cube « normal » (certaines positions ne peuvent être obtenues qu'en démontant le Rubik's Cube).

L'ensemble des positions accessibles du Rubik's Cube est un sous groupe (que l'on appelle groupe du Rubik's Cube : $\mathfrak{R}$) de G ; en effet, on a $\mathfrak{R}$= Gr(R,V,O,W,B,J) c'est à dire que $\mathfrak{R}$ est engendré par les rotations d'un quart de tour dans le sens trigonométrique de chaque face.
On détermine la transcription des mouvements élémentaires manuellement. On remarque que , pour chacun des mouvements élémentaires :
-la permutation des coins et des milieux est impaire
-la somme des indices de rotation des coins est multiple de 3
-la somme des indices de rotation des milieux est paire.
Cela est vrai pour tous les mouvements élémentaires. Pour le Rubik's Cube fait (càd l'identité), les permutations des coins et des milieux ont même parité (1 car identité). Or, n'importe quel mouvement élémentaire modifiera la parité des permutations des coins et des milieux simultanément et de la même façon en composant deux mouvements. Donc, on aura pour toute position accessible du Rubik's Cube, élément de $\mathfrak{R}$: ε(Sigma-M)=ε(Sigma-C).
(ε désigne la signature d'une permutation)
De plus, lorsque l'on compose les indices de rotation, chacun apparaît une et une seule fois dans le produit de deux positions, d'après la façon dont on a construit la loi. Donc, leur somme restera congrue à 0 modulo 2 ou 3 (milieux ou coins).

Les éléments de G dans $\mathfrak{R}$ vérifient donc nécessairement:
-les permutations des coins et des milieux ont même signature (1/2 des éléments de G le vérifient),
-la somme des indices de rotation des coins est multiple de 3 (1/3 des éléments de G le vérifient)
-la somme des indices de rotation des milieux est paire (1/2 des éléments de G le vérifient).

Ces 3 conditions étant indépendantes, on a au plus 1/12 des éléments de G qui sont susceptibles d'appartenir à $\mathfrak{R}$.

- Groupe $\mathfrak{R}$ des positions accessibles : groupe du Rubik's Cube.

Ces 3 critères nécessaires sont suffisants !
Pour démontrer ce résultat, il faut évoquer quelques algorithmes (un 'algorithme' est une suite ordonnée de mouvements élémentaires ayant un effet spécifique sur le Rubik's Cube) basiques permettant de modifier le Rubik's Cube, et issus de l'exploration informatique présentée en (III).

Il existe des algorithmes permettant de:
-faire un 3-cycle de 3 coins coplanaires (ou de 3 milieux coplanaires) sans rien modifier d'autre.
-faire pivoter conjointement 2 milieux d'arêtes adjacentes sans rien modifier d'autre.
-faire pivoter 2 coins adjacents dans deux sens opposés sans rien modifier d'autre.

A partir de ces algorithmes, il est possible de montrer que toutes les positions vérifiant les 3 conditions précédentes sont dans $\mathfrak{R}$.

Lemme : les 3-cycles de S_n engendrent les permutations paires de S_n.

On sait que toute position du Rubik's Cube, c-à-d une position accessible, est caractérisée par l'égalité des signatures de ses permutations de coins et de milieux. On désignera donc par parité d'une position (accessible) la parité de sa permutation de coins ou de milieux.

Soit P une position de G telle que P vérifie les 3 conditions nécessaires d'appartenance à $\Re$, c-à-d que :
$\exists\ \sigma\text{-}c \in S_8$, $\exists\ \sigma\text{-}m \in S_{12}$, $sign(\sigma\text{-}c)=sign(\sigma\text{-}m)$, $\exists\ (IC1,\ldots,IC8) \in \{-1\ ;0\ ;1\}^8$, $IC1+\ldots+IC8 \equiv 0[3]$,
$\exists\ (IM1,\ldots,IM12) \in \{0\ ;1\}^{12}$, $IM1+\ldots+IM12 \equiv 0[2]$, P = {σ-c, σ-m, [IC1, IC2, …., IC8], [IM1, IM2, …, IM12]}

On veut montrer que P est accessible à partir de la position du cube non-mélangé.
Si la position P est paire, alors on part du cube non-mélangé. Si la position est impaire, on part du cube mélangé sur lequel on a effectué un mouvement élémentaire quelconque (c'est une position impaire). Il faudra donc effectuer une série paire de mouvements pour arriver à la position P (lorsque la position de départ et d'arrivée ont même parité, il faut effectuer un mouvement paire pour passer de l'une à l'autre, c'est évident en terme de permutations et de parité : il suffit d'utiliser $\varepsilon(\sigma o \sigma') = \varepsilon(\sigma) * \varepsilon(\sigma')$).

Quoiqu'il en soit, on s'est ramené à devoir effectuer une permutation paire des coins et des milieux d'arêtes pour arriver à la position P (les coins ou les arêtes n'étant pas forcément bien orientés).

On connaît un algorithme qui permet d'effectuer un 3-cycle sur 3 milieux d'arête coplanaires, et un algorithme qui permet d'effectuer un 3-cycle sur 3 coins coplanaires.

Toute permutation paire se décompose en un produit de 3-cycles. Pour faire un 3-cycle sur 3 milieux quelconques, il suffit d'amener ces 3 milieux sur une même face par une manipulation Q (c'est possible et même facile, on connaît un tel mouvement), d'effectuer un 3-cycle A sur ces 3 milieux puis d'effectuer Q^{-1}.(c'est le principe de conjugaison dans un groupe : QAQ^{-1}).
Il est clair qu'une telle manipulation n'affectera que les 3 milieux en question. Ainsi, on est capable d'effectuer un 3-cycle sur 3 milieux quelconques du Rubik's Cube. La démonstration est analogue pour les coins. Etant capable de faire tous les 3-cycles, on peut donc générer toutes les permutations paires des coins ou des milieux.

On peut donc accéder à une position P', analogue à la position P, à l'orientation des coins et des milieux près.
Il faut maintenant montrer que l'on peut orienter les coins (et les milieux) de façon à obtenir la position P

On connaît une manipulation qui fait pivoter sur eux-même deux milieux adjacents.
La position P' est accessible, donc la somme de ses indices de rotations des milieux du cube est paire (on rappelle que c'est une condition nécessaire d'appartenance à $\Re$).
Soit k le nombre de milieux dont l'indice de rotation dans P' ne correspond pas à celui dans P (1 au lieu de 0 ou 0 au lieu de 1). k est forcément paire, sinon la somme des indices de rotation des milieux dans P ne serait pas paire. Donc, il suffit de faire pivoter par deux les milieux en les amenant par une manipulation M l'un à côté de l'autre, puis en effectuant l'algorithme avant d'effectuer M^{-1}. Ainsi, on oriente correctement et 2 par 2 les milieux.
Il est clair que si k était impaire, un milieu unique resterait mal orienté, et la position obtenue ne vérifierait pas (somme des indices de rotation des milieux paire), ce qui serait contradictoire car c'est une condition nécessaire d'appartenance à $\Re$…

La démonstration est du même goût pour les indices de rotation des coins, car on connaît une manipulation qui fait pivoter deux coins adjacents dans des sens opposés.
.
La position P est donc accessible à partir du cube fait. Donc, les 3 conditions nécessaires d'appartenance à $\Re$ sont également suffisantes.

Conclusion:

$\Re$={ {σ-c, σ-m, [IC1, IC2, …., IC8], [IM1, IM2, …, IM12]} / $\sigma\text{-}c \in S8$, $\sigma\text{-}m \in S12$, avec $sign(\sigma\text{-}c)=sign(\sigma\text{-}m)$, $(IC1,\ldots,IC8) \in \{-1\ ;0\ ;1\}^8$ avec $IC1+\ldots+IC8 \equiv 0[3]$ et $(IM1,\ldots,IM12) \in \{0\ ;1\}^{12}$ avec $IM1+\ldots+IM12 \equiv 0[2]$}

Et l'on a: $Card(\Re) = Card(G)/12 = 43252003274489856000 \approx 4.3e19$

II) ANALYSE MATHEMATIQUE DU RUBIK'S CUBE :

- Le groupe du Cube en terme d'applications

On note 'trace' l'application qui associe à un élément de G sa "trace":

$$\text{trace} : G \to \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z}$$
$$X \mapsto \left(\varepsilon(\text{Scoins}(X)) - \varepsilon(\text{Smilieux}(X)), \Sigma_{i\in[|1,8|]} (\text{Icoins}(X))(i), \Sigma_{i\in[|1,12|]} (\text{Imilieux}(X))(i) \right)$$

'trace' est un morphisme de groupes.

On a démontré dans la première partie qu'une position X du Rubik's Cube est atteignable depuis la position de base (identité) si et seulement si trois conditions sont respectées, qui se traduisent par :

$$\varepsilon(\text{Scoins}(X)) - \varepsilon(\text{Smilieux}(X)) = 0$$
$$\sum_{i\in[|1,8|]} (\text{I coins}(X))(i) \equiv 0 \ [3]$$
$$\sum_{i\in[|1,12|]} (\text{I milieux}(X))(i) \equiv 0 \ [2]$$

On en déduit que:

$$\boxed{\mathfrak{R} = \text{Ker trace}}$$

Or, un sous-groupe H de G est distingué ssi H est le noyau d'un morphisme de groupes de G dans un autre groupe G'. On a montré que $\mathfrak{R}$ = Ker (trace) avec 'trace' morphisme groupes. Donc, **$\mathfrak{R}$ est un sous-groupe distingué de G** : $\forall x\in\mathfrak{R}, \forall g\in G, g^{-1}*x*g\in\mathfrak{R}$.

- Quel est le nombre minimal de coups permettant de résoudre n'importe quelle position de $\mathfrak{R}$ en au plus ce nombre de coups ? Définition d'une distance sur $\mathfrak{R}$, μ : diamètre du groupe.

On note μ le nombre minimal de coups permettant de résoudre n'importe quelle position du Rubik's Cube en au plus ce nombre de coups. Une autre formulation serait : μ est le coût maximal de l'algorithme optimal de résolution du Rubik's Cube.

L'algorithme optimal est couramment appelé : **algorithme de Dieu**. Remarque préliminaire : on est sûr qu'il existe un algorithme optimal ; en effet, il existe au moins une série de mouvements optimale (càd minimisant le nombre de coups nécessaires) pour résoudre une position accessible.

Supposons que l'on puisse résoudre n'importe quelle position accessible du Rubik's Cube en au plus μ coups. Alors, cela signifie qu'en inversant les mouvements de résolution (l'inverse d'une série de n mouvements peut se faire en n mouvements), on pourra accéder à toutes les positions du Rubik's à partir du Rubik's fait en au plus μ coups

En 0 coup, on a accès à l'identité (1 position). Lors du premier coup, on a le choix entre 2 quarts de tour pour chacune des 6 faces, ce qui fait donc 12 choix différents. Pour les coups suivants, on aura accès à 11 mouvements différents (tous sauf l'inverse du dernier effectué ce qui annulerait les 2 derniers coups, ce qui ferait alors revenir à l'état du cube 2 mouvements auparavant). Il faut noter que cette étude est peu précise puisqu'elle ne considère pas le fait qu'on peut obtenir la même position avec des mouvements différents. Le seul but est de minorer μ, et de plus il est extrêmement difficile d'éliminer toutes les séries de mouvements équivalentes...

Au final, en n coups au plus, on aura accès au maximum à $1+12+12*11+\ldots+12*11^{(n-1)} = 1+12*\sum_{k=0}^{n-1} 11^k$ positions. Or $\sum_{k=0}^{n-1} 11^k = \frac{11^n - 1}{11 - 1} = \frac{11^n - 1}{10}$

En au plus n coups, on peut donc atteindre au maximum 1+ 12 * $\frac{11^n - 1}{10}$ positions différentes !

On veut donc: $\text{card}(R) \leq \frac{6\,(11^{\mu} - 1)}{5} + 1$

En passant au logarithme: $\frac{\ln\left(\frac{5\,(\text{card}(R) - 1)}{6} + 1\right)}{\ln(11)} \leq \mu$

Or, card($\Re$) ≈ 4,3e19. Application numérique: μ≥18,7.
Le premier entier naturel vérifiant cette inégalité est μ=19. Nous avons donc obtenu de façon simple une minoration appréciable de μ. **Ainsi, est on sûr que l'on ne pourra pas résoudre toutes les positions du Rubik's en moins de 19 coups.**

Cependant, on peut minorer plus finement μ. Chaque mouvement que l'on effectue (quart de tour d'une des 6 faces dans le sens trigonométrique ou horaire) est caractérisé par une permutation impaire des coins et des milieux. Donc, tout mouvement élémentaire modifie la parité de la position du Rubik's Cube sur lequel on l'effectue. Cela provient du fait que la signature d'un produit de permutations est égal au produit des signatures de ces permutations. Ainsi, si on part d'une position de parité ε , et que l'on effectue à partir de là n mouvements élémentaires, la parité de la position finale que l'on obtiendra après ces n coups sera de: $(-1)^n \varepsilon$. Cette remarque formelle traduit juste le fait que, chaque mouvement élémentaire induisant une permutation impaire des coins et des milieux, la parité de la position s'en trouve de fait modifiée. Une conséquence directe de ce constat est que pour arriver au cube fait (l'identité, paire), il faudra effectuer un nombre pair de mouvements si position de départ est paire, et un nombre impaire sinon. (On a déjà utilisé cette remarque dans la première partie, pour déterminer $\Re$). **Ainsi, résoudre une certaine position requiert forcément un nombre de coups dont la parité est celle de la position en question.**

On va s'intéresser aux permutations impaires et aux positions impaires du cube. On va également éliminer les doubles mouvements qui se recoupent. Ensuite, il existe 3 couples de faces opposés: Rouge-Orange, Bleu-Blanc(W) et Jaune-Vert. Pour chacun de ces couples, les deux faces étant opposées, il est indifférent de tourner l'une ou l'autre en premier: ainsi, R-O et O-R sont équivalents, c'est évident (car les permutations des coins et des milieux des deux faces sont à supports disjoints). On peut obtenir un mouvement redondant en tournant chacune des faces dans le sens trigonométrique ou horaire (R-O, r-O, r-o et R-o). Il y aura donc 4 mouvements redondants par couple de faces(R-o <=> o-R, etc.), soit au total 12 mouvements redondants pour les 3 couples. On montre qu'il y en a exactement 18. On montre ainsi que : **μ≥21**.

La borne supérieure peut s'obtenir empiriquement: à partir des algorithmes de résolution du Rubik's, on calcule le coût "au pire" de chacune des étapes de résolution. J'ai trouvé ainsi : **μ≤ 200**.

La valeur de μ est un problème encore ouvert ; cependant, il a été informatiquement démontré que μ≥26 (en montrant qu'une certaine position ne pouvait être atteinte en moins de 26 coups). De plus, un mathématicien de l'Université de Tennesse, Morwen Thistlethwaite, fit des recherches dans les années 80 et établit un algorithme extrêmement complexe de résolution, qui ne demande que 52 coups au plus pour résoudre n'importe quelle position accessible, et qui utilise des sous-groupes particuliers du Rubik's Cube. Ce nombre a depuis été ramené à 42 ; ainsi, des études complexes ont permis de démontrer que : μ≤ 42.

On souhaite maintenant définir une distance sur le groupe $\Re$. Soient a et b 2 éléments de $\Re$. On note A={ensemble des manipulations qui permettent de passer de a à b}. A est non vide car les 2 positions sont accessibles : il suffit donc de revenir à l'identité à partir de a puis d'aller jusqu'à b à partir de l'identité. On note B={ensemble des longueurs des manipulations de A}. B est un ensemble non-vide d'éléments de IN, et admet donc un plus petit élément, que l'on appellera d(a,b). Une manipulation dans A de longueur d(a,b) est appelée manipulation optimale pour passer de A à B.

On définit une distance entre deux positions du Rubik's Cube accessibles comme le nombre minimal de mouvements élémentaires permettant de passer d'une position à l'autre

On montre facilement que c'est une distance. **μ s'appelle le diamètre du groupe pour cette distance** que l'on a défini. Soient a et b 2 positions. La distance entre $(a^{-1})*b$ et l'identité est d'au plus μ. Soit L une manipulation optimale qui permette de passer de l'identité à $(a^{-1})*b$. L se fait en au plus μ mouvements élémentaires ; or, en effectuant L sur la position a, on obtient $a*(a^{-1}*b)=b$. Donc, d(a,b)≤μ.

La distance entre 2 positions est donc inférieure ou égale à μ.

- Groupes isomorphes à un sous-groupe du Rubik's Cube :

Théorème de Cayley : Soit (G,.) un groupe fini de cardinal n. Alors G est isomorphe à un sous-groupe de S_n (groupe des permutations de [1;n]). (démo : on utilise l'injectivité de la fonction g(x) : y → y*x).
Il existe des permutations des 12 milieux dans $\mathfrak{R}$; On construit un sous-groupe de $\mathfrak{R}$ isomorphe à S12 ; on injecte Sn pour n<=12 dans S12, et **ainsi tout groupe de cardinal inférieur ou égal à 12 est isomorphe à un sous groupe du Rubik's Cube**. Ainsi, il est intéressant de constater qu'il est possible de représenter concrètement des structures algébriques complexes tels des groupes.

- L'ordre maximal pour un élément de $\mathfrak{R}$ est 1260. $\forall x \in G, x^{166320} = Id$

On constate que pour maximiser l'ordre d'une position, il faut maximiser le PPCM de la longueurs des cycles intervenant dans le décomposition en cycles des permutations des coins et milieux Si l'on compose à son tour x' 6 fois avec lui-même, on multiplie d'autant les indices de rotation des cubes coins et des cubes arêtes modulo 2 ou 3. On démontre que 1260 est l'ordre maximal atteignable.

Par ailleurs, soit x un élément de G. Smilieux(x) et Scoins(x) peuvent se décomposer en cycles de longueur comprise entre 2 et 12 (inclus).

Soit n = PPCM(2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12) = 27720

Si on compose 'x' n fois, tous les cycles de Smilieux(x) et de Scoins(x) sont composés n fois et donnent nécessairement l'identité. Donc, x^n est une position où tous les cubes coins et tous les cubes milieux sont en place (sans être forcément tournés dans le bon sens). Cela implique :

$$(x^n)^6 = Id, \text{ et donc : } x^{6 * 27720} = x^{166320} = Id$$

Ainsi l'ordre de tout élément de $\mathfrak{R}$ est inférieur à 1260, et divise 166320 :

$\forall x \in \mathfrak{R}$, Card(<x>) ≤ 1260

$\forall x \in G, x^{166320} = Id$ et $x^{-1}=x^{166319}$.

- Le centre du groupe est réduit à 2 éléments :

Lemme: Soit n un entier strictement supérieur à 2. Le centre de Sn est { Id }.
Ce lemme permet de montrer qu'il n'y a que 2 positions dans $\mathfrak{R}$ qui commutent avec toutes les autres :
L'identité et la position analogue à l'identité avec tous les milieux d'arêtes pivotés de 180° (Superflip)

Le centre du groupe $\mathfrak{R}$ est : $C(\mathfrak{R})$ = {Id, Superflip}.

- $\mathfrak{R}$ est engendré par 5 mouvements élémentaires seulement. $\mathfrak{R}$ est engendré par 2 éléments seulement

Un des 6 mouvements éléméntaires est superflu (et moins que 5 ne convient pas car moins de 5 mouvements laisserait invariants certains cubes élémentaires). C'est-à-dire que $\mathfrak{R}$ = Gr(W,R,J,V,B) par exemple.

Cependant, encore plus surprenant : $\mathfrak{R}$ est engendré par deux éléments seulement

La démonstration très technique se fait en construisant les deux mouvements engendrant $\mathfrak{R}$, puis en montrant qu'ils engendrent une position quelconque de $\mathfrak{R}$, donc $\mathfrak{R}$; cette démonstration fonctionne cependant comme la démonstration du théorème suivant, tout en utilisant le principe de conjugaison :
S_n est engendré par deux éléments (une transposition et un cycle de longueur (n-1) idiones).

III) L'INFORMATIQUE COMME OUTIL D'EXPLORATION

- Mise en place de la structure de groupe du Cube sous CAML

J'ai eu recours au langage CAML pour mettre en place la structure de groupe du Rubik's Cube sous une interface informatique ; cela a permis d'effectuer certains calculs et vérifications extrêmement rapidement, et de mener des recherches inconcevables sur papier. L'ordinateur sous lequel j'ai travaillé était équipé d'un Pentium® IV cadencé à 1,4GHz et de 512Mo de mémoire vive, tournant sous Windows XP™.

Préliminaire : une position du cube sera codée par un tableau de 4 tableaux d'entiers (type 'int vect vect' défini comme le type 'rubik').

(*c_comp, une sous-fonction de 'x', compose deux tableaux de permutations des coins dans S8, dans l'ordre inverse: sc2 o sc1*)
(*m_comp, un sous-programme de 'x', compose deux tableaux de permutations des milieux dans S12, dans l'ordre inverse: sc2 o sc1*)
(*ic_comp, un sous-programme de 'x', compose deux tableaux d'indices de rotation des coins ('im1' et 'im2'), en utilisant le tableau des permutations ('sig') des coins, qui est nécessaire (cf la formule de composition de 2 positions du cube) et fait en sorte que le résultat soit dans [-1;1] pour plus de lisibilité *)
(*im_comp, un sous-programme de 'x', compose deux tableaux d'indices de rotation des milieux (im1 et im2), en utilisant le tableau des permutations des milieux ('sig'), qui est nécessaire (cf la formule de composition de 2 positions du cube) *)
(* Une position du cube est codée sous forme de "int vect vect", un tableaux contenant 4 tableaux d'entiers. On définit le type rubik qui désigne une position du cube *)
Programme principal de composition : (* 'x' compose deux positions du cube en appliquant la formule de composition*)
(* on rend la fonction de composition infixe pour plus de commodité*)
Données fondamentales indispensables pour la suite :
(*V, W, O.... représentent les mouvements élémentaires d'un quart de tour trigonométrique d'une face de chaque couleur. Je les ai recherchés manuellement en regardant comment la rotation de chaque face affecte les cubes élémentaires (coins et milieux), avec mes notations et conventions*)
(*compo_list compose dans l'ordre une série de mouvements élémentaires contenus dans une liste*)
(*r, w, o.... représentent les mouvements élémentaires d'un quart de tour horaire d'une face de chaque couleur ; R2, W2, O2... représentent les mouvements élémentaires d'un demi-tour d'une face de chaque couleur ; id représente le cube fait, càd l'identité*)
(*cette fonction donne l'ordre d'une position, qui est inférieur ou égal à 1260 pour une position accessible.*)
Un exemple frappant de l'utilité d'un ordinateur : ce programme peut composer en quelques millièmes de secondes un même mouvement des milliards de fois, ce qui prendrait à la main un temps… considérable !
(*'compo' compose de manière optimisée et par récurrence 'mouvement' 'n' fois ; le temps d'exécution est en O(ln n) où n est le nombre de répétitions. Le principe utilisé est analogue à celui de l'exponentiation rapide*)
(*ce programme détermine si une position donnée 'pos' engendre un sous groupe de G d'ordre 1260*)
(*ce programme détermine si une position donnée appartient au U-Groupe, càd si les 2 couronnes supérieurs correspondent à celle de l'identité. On ne s'intéresse pas aux positions triviales. *)
Désormais, avec les fonctions et les données précédentes, le Rubik's Cube est implémenté sous une interface informatique et l'on va pouvoir utiliser la puissance de calcul de l'ordinateur.

- Quelques programmes d'intérêt mathématique

(*cette fonction donne l'inverse d'une position*)
(*sign donne la signature d'une permutation de Sn représentée par sa matrice ; on utilise la formule avec la décomposition en cycles disjoints*)
(* 'cycles' donne la décomposition en produit de cycles à supports disjoints d'une permutation de Sn représentée par sa matrice.*)
(* 'cycles2' est une version en O(n) mais plus lourde du point de vue de la programmation ; on marque les valeurs par lesquelles on est déjà passé. La même astuce peut être utilisé dans 'signature' mais ne présente pas d'intérêt *)

Cette fonction a en plus un intérêt pratique pour vérifier qu'une position aléatoire est accessible, cf. partie IV
(*signature donne la signature d'une position du Rubik's Cube sous la forme d'une liste de 3 entiers ; cette position est accessible ssi sa signature vaut [0;0;0]*)
(*'parité' donne la parité d'une position accessible, 1 si elle est paire, -1 si elle est impaire*)

- Compression des positions

Il s'est avéré nécessaire de comprimer les données lors des recherches de millions de positions que l'on devait ensuite stocker en mémoire. Voici l'ensemble des fonctions (très techniques et dont seule la finalité est importante) permettant de compresser et décompresser les positions ; le principe est de transformer la liste de 4 matrices d'une position en une liste de 4 entiers naturels qui prennent moins de place en mémoire. On obtient ainsi un facteur 10 de compression des données (on aurait encore pu améliorer ce facteur en travaillant bit à bit par exemple).

(* 'rang' calcule le rang dans l'ordre lexicographique d'une permutation des coins, dans S8, càd produit une bijection de S8 dans [|0;8!-1|]*)
(* 'rang2' calcule le rang dans l'ordre lexicographique d'une permutation des milieux, dans S12, càd produit une bijection de S12 dans [|0;12!-1|]*)
(* 'enbase2' convertit un tableau d'entiers (0 et 1) de longueur 12 en un entier dont l'écriture en binaire est la suite de 0 et 1 du tableau *)
(* 'enbase3' convertit un tableau d'entiers (0, 1 et -1 que l'on transforme en 2) de longueur 8 en un entier dont l'écriture en base 3 est la suite de 0, 1 et -1 (pour 2) du tableau *)
(* debase2 est la fonction réciproque de la fonction 'enbase2', il fournit le tableau de longueur 12 d'entiers 0 ou 1 correspondant à l'entier qu'il reçoit en paramètre*)
(* debase3 est la fonction réciproque de la fonction 'enbase3', il fournit le tableau de longueur 8 d'entiers 0, 1 ou -1 correspondant à l'entier qu'il reçoit en paramètre*)
(*cette fonction est la bijection réciproque de 'rang2' et donne la permutation de S12 correspondant à un certain rang*)
(*cette fonction est la bijection réciproque de 'rang' et donne la permutation de S8 correspondant à un certain rang*)
(*supertab est un tableau contenant la correspondance entre une permutation de S8 et son rang ; il est plus rapide de calculer une fois pour toute ce tableau plutôt que d'utiliser la fonction inverse8 à chaque fois ; pour des raisons de mémoire, cela n'est pas possible avec S12 pour les permutations des milieux*)
type zip == int vect;;
Les 2 fonctions-clef de compression/décompression de positions :
(*cette fonction contracte une position du Rubik's Cube, càd la transforme en une liste de 4 entiers qui est moins gourmande en mémoire*)
(*cette fonction détracte une position du RUbik's CUbe, càd transforme la liste de 4 entiers en une position que l'on peut composer facilement*)

- Recherche des mouvements que l'on peut atteindre en n coups exactement : Pos(n)

Par la suite, on notera Pos(n) le nombre de positions atteintes en exactement n coups (pas moins), et pos(n)=Card(Pos(n))
(un coup est un mouvement élémentaire appliqué à une position).
Ainsi, on a : pos(0)=1 et pos(1)=12 et pos(2)=114.
(*un exemple de fonction naïve de recherche de positions*)
Ce programme utilise la « force brute » de l'ordinateur en testant toute les combinaisons de mouvements à partir d'une certaine position du cube pour arriver à une position vérifiant certaines conditions ; ces conditions sont déterminées par un sous-programme dont on fournit le nom en argument.
(* force_brute prend en argument un programme 'condition' de type (rubik -> bool) sélectionnant certaines positions particulières (par exemple in_1260 ou in_ugroup), et détermine toutes les positions du cube engendrées

par au plus 'profondeur' mouvements élémentaires (le 1er mouvement élémentaire effectué sur 'position' est différent de 'interdit') sur 'position' vérifiant 'condition' *)

Eléments de $\mathfrak{R}$ d'ordre 1260 obtenus en au plus 6 coups à partir du cube fait :

Nous avons utilisé le programme force_brute pour trouver tous les éléments d'ordre 1260 à une distance au plus 6 du cube fait. Pour cela, nous avons lancé les commandes :
force_brute in_1260 w 5 W et nous avons trouvé tous les mouvements à un isomorphisme près. Le mouvement dit « de Butler » n'est donc qu'un mouvement générique d'une famille qui peut être obtenue informatiquement.
(*fonctions plus évoluées*)
Nous allons par la suite utiliser des méthodes plus évoluées mettant en œuvre 2 techniques : la compression de données pour palier au problème (bien réel !) de la mémoire, et le tri des positions selon leur permutation des coins pour plus de rapidité.
Pour donner un ordre d'idée, avec une fonction naive telle 'force_brute', il est inconcevable du point de vue de la rapidité de mener des recherches de positions avec une profondeur supérieure à 8. La recherche de positions atteintes en n coups ne peut se faire pour n>5 par cette méthode, en un temp raisonnable ; il était donc nécessaire d'élaborer de nouvelles techniques.
L'objectif des programmes qui suit est principalement de recenser le nombre de positions (distinctes) du Rubik's Cube que l'on peut atteindre en n mouvement exactement (et pas en moins !). La difficulté réside dans le fait qu'il faut stocker et comparer un nombre considérable de positions. Nous avons mené les recherches jusqu'à n=9 inclus ; n=10 est encore possible avec ce principe, en utilisant une compression meilleure et avec un peu plus de mémoire. Pour n=11, il faudrait envisager une méthode plus rapide et moins gourmande en mémoire (c'est à dire une méthode radicalement différente).

Le programme suivant est essentiel pour la recherche ; il énumère à partir du rang n-1 toutes les positions que l'on peut atteindre au rang n en vérifiant que ces positions n'ont pas pu etre atteintes en moins de mouvements.
On fait la remarque suivante : en effectuant un mouvement élémentaire à partir d'une position obtenue en n coups exactement, on est évidemment sur que la nouvelle position s'obtient en au plus n+1 coups, mais on sait aussi qu'elle s'obtient en n-1 ou en n+1 coups (pas en n pour des raisons évidentes de parité déjà évoquées, pas en n-3 ni en moins car sinon la position de départ serait atteignable en moins de n mouvements, ce qui serait contradictoire car elle est atteinte en exactement n mouvements et pas moins). Ce principe permet de limiter au rang n+1 et n-1 les vérifications qui déterminent si une position a déjà été atteinte.
Ensuite, on classe les positions dans un tableau selon le rang de leur permutation de coins (on rappelle que rang est une fonction bijective qui numérote les éléments de S8, groupe des permutations à 8 éléments). La fonction rang utilise l'ordre lexicographique sur les matrices de permutations des coins.
Ce programme utilise moins de mémoire en travaillant sur des positions compressées, et va plus vite en triant ses positions et en ne recherchant la position qu'au rang n+1 et n-1.
On peut remarquer que le programme a une structure très intuitive.
Note : le fait de classer les positions selon le rang de leur permutation de coins rend obsolète le premier nombre constituant la 4-liste codant une position compressée ; en effet, on a accès à ce nombre en déterminant l'emplacement de la position dans la liste ; cependant, cette astuce ne fait gagner qu'un petit facteur mémoire (diminution d'environ 25% de la mémoire utilisée) qui est à comparer au facteur 10 (environ) en mémoire pour passer du rang n au rang n+1.

let nb_positions n =
let perm_coins () =
Le programme perm_coins permet de montrer que toutes les permutations impaires des coins sont atteintes en 7 mouvements exactement, donc en 7 mouvements au plus ; dès lors, considérons une permutation paire des coins S. Soit s la permutation des coins induite par le mouvement H (haut). La permutation impaire (s^-1)oS est atteinte en au plus 7 coups ; soit P une position accessible possédant cette permutation de coins (on est sûr qu'il existe une telle position). En effectuant H sur cette position, on obtient une position (en au plus 8 coups…) dont la permutations des coins est : so(s^-1)oS = S.
Conclusion : on peut atteindre une permutation quelconque des coins en au plus 8 mouvements.

- Positions isomorphes. Théorèmes concernant les morphes.

Recherche optimisée de Pos(n) (minimiser la mémoire requise et le temps d'exécution) ; facteur de croissance, hypothèses sur la forme de la courbe. Algorithme de Dieu pour le Rubik's Cube 2x2x2.

Définition de positions isomorphes : 2 positions sont dites isomorphes ssi elles peuvent etre obtenues en effectuant la même série de mouvement (ou l'inverse de cette série de mouvements) sur le Cube, soit sur des faces différentes et en changeant l'orientation du Cube, soit de façon symétrique (comme dans un miroir).

De par la définition, et sachant qu'il existe 24 orientations spatiales d'un cube, il existe pour une position donnée au plus : 2x2x24=96 isomorphes.
Ainsi, les 12 mouvements élémentaires engendrent des positions isomorphes.

Intuitivement, cela correspond aux positions « qui se ressemblent » (en changeant l'orientation du cube, ou en le regardant dans un miroir), ou à leurs inverses.

Remarque : « est isomorphe à » est une relation d'équivalence.

Définition : le morph de la position P est le nombre de positions isomorphes à cette position P. Il est compris entre 1 et 96.

Théorème : si une position est atteinte en n coups exactement, alors une position qui lui est isomorphe est également atteinte en n coups exactement.

L'intérêt de ce théorème est que lorsque l'on connaît une position, on en connaît immédiatement une famille, ce qui va faciliter notre travail de recherche.

Théorème: il existe exactement 8 positions ayant un morph de 1, et elles sont toutes paires ;
Question : quelles sont les valeurs possibles de la fonction morph pour une position accessible ? Pour une position quelconque ?
Nous n'avons pas trouvé de réponse satisfaisante à cette question. 1,3,6,12,24,48,96 sont en tous cas possibles.

La valeur moyenne du morph pour une position tirée aléatoirement vaut 96.

Programmation en CAML :
Commentaires sur cette série de fonctions : pour déterminer l'ensemble des positions isomorphes à une position donnée, on procède comme suit : on détermine les positions correspondant aux 24 façons d'orienter le cube dans l'espace, on calcule le symétrique de chacune des positions par rapport à la couronne du milieu, puis l'inverse des 48 positions déjà obtenues, soit au total 96 positions ; on élimine les doublons.
Pour déterminer l'inverse d'une position on construit une série de programmes performants utilisant la formulation mathématique de l'inverse d'une position.
Pour déterminer le symétrique d'une position par rapport à la couronne du milieu, on permute de façon adéquat les numéros des pièces et l'on modifie les indices en se référant aux conventions prises.
Enfin, pour déterminer les 24 orientations possibles du Cube, on utilise le principe de conjugaison par des positions non-accessibles (cela est licite car R est un sous-groupe distingué de G) ; ces positions correspondent à faire un 4 cycle de 4 centres coplanaires…

```
(*bloc de fonctions pour inverser rapidement une position*)
 (* 'spotH', 'spotG' et 'spotF' sont 3 positions non accessibles du Rubik's Cube que l'on utilise avec le principe de conjugaison pour trouver les différentes orientations du cube*)
 (*on définit ici les 24 orientations spatiales possibles du Rubik's Cube*)
 (*on calcule les orientations opposées pour plus de rapidité par la suite dans les calculs*)
 (*cette fonction 'symétrie' renvoie le symétrique d'une position par rapport au plan (spotG) *)
 (* 'listomorph' renvoie la liste des positions isomorphes 2 à 2 distinctes d'une position *)
 (* 'morph' renvoie le nombre de positions isomorphes 2 à 2 distinctes d'une position ; le morph d'une position vaut au plus 96 et au moins 1 *)
 (* cette fonctions détermine si 2 positions sont isomorphes *)
 (* cette fonction détermine le plus petit élément d'une liste dans l'ordre total lexicographique. Elle est polymorphe *)
```

La fonction suivante est importante car elle permet beaucoup plus de rapidité dans la recherche d'isomorphisme à grande échelle : on associe à chaque famille d'isomorphes un mouvement de référence unique qui caractériste complètement la famille. On choisit la position dont la transcription compressée est minimum du point de vue de l'ordre lexicographique !

```
(* 'pos_ref' renvoie la position de référence associée à un représentant d'une famille d'isomorphe, et le nombre d'éléments de la famille d'isomorphes*)
```

L'intérêt du programme précédent est qu'il optimise la vérification pour savoir si l'isomorphe d'une position a déjà été atteint…
Le programme suivant est la version définitive de recherche de positions ; il divise par près de 96 la mémoire utilisée, en ne considérant pour chaque mouvement qu'un élément de la famille des 'au plus 96' isomorphes ! Le facteur est proche de 96 car la majorité des positions obtenues en n coups pour n assez grand (5, 6 ou 7) ont un morph de 96. (la valeur moyenne du morph pour n=6 par exemple vaut à peut près 86,5 , pour n=8 elle vaut 92, et cette valeur croît avec n).
Du point de vue de l'affichage, qui donne une bonne indication de l'avancement des recherches et qui est donc très important pour estimer le temps d'exécution du programme en fonction du pourcentage des recherches effectuées, il a été nécessaire de mélanger l'ordre dans lequel les permutations des coins sont analysées(en effet, l'ordre lexicographique favorise les positions dont la permutation des coins a un rang petit, d'où une inhomogénéité : le programme est plus lent lorsqu'il est dans les zones où le rang est faible càd en début de liste ; pour remédier à ce probleme, on parcourt la liste dans tous les sens en utilisant le principe de générateur dans (Z/nZ,+))
A noter que cette fois, les éléments ayant même permutation de coins sont triés en leur dans l'ordre lexicographique pour plus de rapidité.

Pour résumer, la fonction finale de recherche de positions utilise les astuces suivantes :
-compression des positions (facteur 10 pour la mémoire)
-une position atteinte en n coups exactement combinée avec un mouvement élémentaire donne une position atteinte en n°1 ou n-1 coups exactement (faible gain de temps)
-regroupement des positions en fonction de leur permutation des coins(gain de temps considérable, facteur 100 probablement)
-classification de chaque groupe dans l'ordre lexicographique (faible gain de temps)
-utilisation des positions isomorphes pour diminuer la mémoire utilisée, et introduction d'un mouvement de référence (facteur proche de 100 pour la mémoire).
-processus de parcours des permutations des coins idoine pour que l'affichage soit homogène (on a ainsi un ODG du temps d'exécution du programme après quelques minutes en extrapolant) (on utilise un élément générateur grand de Z/(8!)Z,+)

Si l'on fait un bilan, on a divisé par près de 1000 la mémoire utilisée et on va 100 fois plus vite par rapport à une version naïve d'un programme de recherche. Les recherches pour n=9 ont demandé à CAML 500MO de mémoire vive et 40h de calcul ; il aurait fallu avec la version naïve près de 500GO de mémoire et aussi 6 mois de calcul… Sans commentaires.

```
let no_x n =
(*ce programme tire au sort une position accessible du Rubik's Cube*)
 (* près de 100% des positions aléatoires ont un morph de 96 ; nous avons tiré un million de positions aléatoirement, elles avaient toutes un morph de 96*)
```

Résultats obtenus avec le programme 'no_x' précédent:

n	pos(n)
0	1
1	12
2	114
3	1,068
4	10,011
5	93,840
6	878,880
7	8,221,632
8	76,843,595
9	717,789,576

On peut remarquer qu'en première approximation, 10^n donne un bon ordre de grandeur pour po(n).
On appelle facteur de croissance au rang n et l'on note fc(n) le rapport pos(n+1)/pos(n).
fc(6) = 9 ;355 ; fc(7)=9,347 ; fc(8)= 9,347. (ces nombres ont été arrondis). Une étude de ce rapport permet de formuler des hypothèses sur le comportement à court terme de pos(n). Ainsi est-il raisonnable de penser que la valeur suivante vaut par exemple : pos(10) ~ pos(9)*9,346 ~ 67 milliards (l'approximation est très bonne, ce nombre vaut en effet 67 milliards et des poussières de millions).
Cependant, nécessité pour l'égalité paire/impaires qu'il y ait une chute de la croissance;gain par rapport à 12*11^(n-1), hypothèses sur la forme de la courbe.

- Le U-Groupe, un sous-groupe particulier de $\mathfrak{R}$.

(Utilisation de la fonction 'force_brute' → algorithmes de résolution du Rubik's Cube)

> On appelle U-Groupe et l'on note U le sous-groupe du Rubik's Cube constitué par l'ensemble des transcriptions des mouvements qui ne modifient que la face du bas.

Cet ensemble est d'importance pratique : son étude permet d'obtenir des algorithmes utiles à la résolution du Rubik's Cube. Nous avons ainsi retrouvé les algorithmes que nous utilisons pour remonter le Rubik's Cube, tels que l'algorithme dit "final". Le petit nombre d'algorithmes trouvés en un nombre raisonnable de coups (8 au plus) suffit à montrer que la résolution du Rubik's Cube au hasard est quasiment impossible sans une étude théorique préalable, ou au moins une analyse judicieuse de son fonctionnement.
#force_brute in_ugroup g 7 G ;;
On est maintenant capable à partir de ces mouvements de faire (expression avec la 2nde notation pour plus de commodité dans la réalisation pratique):
-Un 3-cycle de 3 coins de la face du bas (en changeant l'orientation de certaines pièces de la dernière face) (gPDpGPdp), en 8 coups.
-Un 3-cycle des milieux de la face du bas (en changeant l'orientation des 4 coins et de deux milieux de la dernière face) en composant 2 fois l'algorithme « le plus court » (gapAPapAPG), en 10 coups.
-Pivoter deux coins adjacents de la face du bas sans rien modifier d'autre, en 16 coups, en composant un 3-cycle sur les coins avec rotation de deux coins et un 3-cycle avec rotation de 3 coins : (gaGfgAGFgaDAGadA)
-pivoter deux milieux adjacents de la face du bas sans rien modifier d'autre

> En modifiant l'orientation du Cube et en utilisant le principe de conjugaison dans un groupe, on est capable d'effectuer un 3-cycle sur 3 coins ou milieux quelconques du Rubik's Cube, et de faire pivoter simultanément et dans des sens opposés 2 coins ou milieux quelconques du Rubik's Cube.

Ainsi, on sait désormais résoudre le Rubik's Cube…

CONCLUSION :

-Un objet mathématique complexe : on, peut munir le Rubik's Cube d'une structure de groupe et démontrer à l'aide des mathématiques des propriétés remarquables.

- Somme toute, le Rubik's Cube reste un problème largement ouvert. Ainsi, nul n'est encore parvenu à décrire un algorithme optimal de résolution du cube ni la valeur de µ: on s'est contenté de nommer un tel objet hypothétique "algorithme Dieu", à défaut, sans doute, de pouvoir le baptiser du nom de son inventeur. Il y a plus : le Rubik's Cube peut se généraliser à un objet de taille quelconque, dans un espace de dimension quelconque ; ainsi, mathématiquement, on peut parler d'un Rubik's Cube NxNxN en dimension p. (généralisation purement virtuelle et mathématique).

-A première vue, le Rubik's Cube classique que nous avons étudié ici peut sembler bien réduit ; et pourtant, si l'on pouvait examiner un milliard de positions de ce cube par seconde, ce qui est aujourdhui complètement irréaliste, il faudrait encore trois siècles pour le parcourir entièrement - ce qui n'est pas moins irréaliste.

-Utilisation de l'informatique : permet d'explorer le Rubik's Cube

-Le travail récent de Korf (1997) et l'IDA*. Intelligence artificielle, permet de résoudre rapidement une position en un nombre optimal de coups.

ANNEXE : VOCABULAIRE

Cube désigne le Rubik's cube lui-même.

Position du cube : c'est un agencement du cube que l'on peut obtenir en démontant le Rubik's cube puis en le remontant de façon désordonnée ou aléatoire. Le cube obtenu constitue une position quelconque du cube. La position correspondant au cube fait est appelée 'identité'. Les positions que l'on peut atteindre par un mouvement quelconque à partir du cube fait sont dénommées ***positions accessibles*** ; les positions restantes constituent les ***positions inaccessibles*** à partir d'un cube fait par des suites de mouvements élémentaires.

On prend garde à distinguer, dans l'étude de la structure de groupe, le ***groupe du cube, G*** (regroupe l'ensemble des positions théoriques du cube) et le ***groupe du Rubik's Cube, $\mathfrak{R}$*** (qui regroupe seulement l'ensemble des positions accessibles pratiquement, sans démonter le cube, à partir du cube fait, càd que $\mathfrak{R}$ = Gr (W,V,B,O,R,J)).

On désignera par ***cube fait*** (ou ***non-mélangé*** ou encore ***identité***) un cube non mélangé, comme celui de la figure 1.1 (page suivante).

cube élémentaire ou ***petit cube***: c'est l'une des 26 pièces mobiles constituant le cube (27 – 1 pour le centre dans lequel se loge le mécanisme de rotation)

Une ***face*** est un ensemble de 9 cubes élémentaires coplanaires. Il existe 6 faces, chacune caractérisée par une couleur, celle du cube élémentaire qui se trouve en son milieu. Nous appellerons les 6 faces W (blanc pour « white » en Anglais), B (bleue), R (rouge), V (vert), O (orange), R (rouge).

Une ***facette*** est une portion d'un cube élémentaire sur laquelle est collée une vignette. Il existe 54 facettes en tout (9 de chacune des 6 couleurs)

Coin (ou ***cube sommet***): c'est tout cube élémentaire commun à 3 faces, portant 3 facettes. Il en existe 8 sur le cube, chacun caractérisé par les couleurs de ces 3 facettes (2 à 2 distinctes) et situé sur un sommet du cube. Ex : RWV désigne l'unique coin portant les couleurs rouge, blanc et vert.

On appelle ***centre*** d'une face le cube élémentaire qui se trouve en son centre et qui, par sa couleur donne son nom à cette face. Un centre a une position fixée par rapport aux autres centres (il peut cependant tourner sur lui-même. Un cube élémentaire situé au milieu d'une arrête du cube est appelé ***milieu (***ou ***cube arête)*** et est caractérisé par la couleur de ses 2 facettes.

On appelle ***mouvement élémentaire*** (ou ***coup***) toute rotation d'une des 6 faces d'un quart de tour dans le sens trigonométrique ou horaire.

Mouvement (ou manipulation) : c'est une suite ordonnée de mouvements élémentaires.

Algorithme : mouvement générique (à un isomorphisme près) ayant un effet spécifique sur le Rubik's Cube.

Transcription : c'est une bijection entre l'ensemble des positions quelconques du cube et un ensemble d'objets mathématiques explicités dans la 1ère partie.

Parité d'une position (accessible) : c'est la parité de la permutations des coins d'une position accessible.

Décomposition en mouvements élémentaires d'une position : suite de mouvements élémentaires qui, effectués dans l'ordre sur l'identité, donnent cette position.

Couronne : c'est un plan du cube parallèle à une face (chaque couronne contient donc 9 cubes élémentaires, exceptée la deuxieme couronne, au milieu, qui n'en contient que 8).

- BIBLIOGRAPHIE :

Site Internet « http://www.geocities.com/jaapsch/puzzles/theory.htm » pour l'existence d'un encadrement de μ entre 26 et 42, pour l'existence d'un mouvement dit « de Butler » d'ordre 1260, pour l'appelation de « superflip », ainsi que pour le travail du mathématicien M. Thistlethwaite.