

CYCLE L (L3 2013)

Aide à la décision TP N°1

But: Appliquer le A* à un cas concret – recherche d'heuristiques

SUJET

Réaliser en C un solveur de Rubik's Cube en utilisant l'algorithme A*.

Elaborer, sur la base d'un travail de recherche personnelle, plusieurs heuristiques et comparer l'efficacité relative de celles-ci.

Prendre comme heuristique de référence (peu performante) le dénombrement des facettes mal placées (cf. l'exercice d'application du A* au Taquin vu en TD).

CONSEILS POUR LA MODELISATION DES ETATS ET DES TRANSITIONS

Structure algébrique

Un Rubik's Cube est composé de 6 faces chacune composée de 9 facettes.

Ce cube peut être transformé à l'aide de 6 opérations de base que l'on peut enchaîner.

L'ensemble de ces 6 opérations est le générateur d'un groupe d'action qui est l'ensemble des transformations licites du Rubik's Cube à partir d'une position initiale.

L'ensemble des positions possibles du Rubik's Cube est donc le cardinal de ce groupe d'action.

Nous distinguons ci-dessous, pour des raisons pratiques, un ensemble d'opérations élémentaires dont le coût est unitaire du générateur évoqué ci-dessus.

Convention, notations

Une convention est un arbitraire nécessaire pour fixer un référentiel formel commun.

Commençons par associer un index à chacune des 6 faces du cube dans sa version résolue.

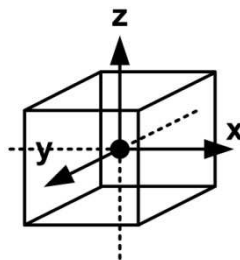
Les 6 faces sont respectivement colorées en blanc, bleu, rouge, orange, vert, jaune. On associe respectivement à ces 6 couleurs les identifiants 1 à 6.

Repère

Le cube est tenu face à soi, face blanche au premier plan, face jaune à l'arrière plan, face rouge au dessus, face orange en dessous, face bleue sur le côté gauche, face verte sur le côté droit.

En dérivent trois axes de rotation perpendiculaires à chacun des couples de faces opposées, traversant leur facette centrale :

- l'axe x horizontal, orienté de gauche à droite, qui traverse les faces bleue et verte,
- l'axe z vertical, orienté de bas en haut, qui traverse les faces orange et rouge,
- l'axe y de profondeur, orienté de l'arrière à l'avant, qui traverse les faces jaune et blanche,



Opérations

Définissons à présent les 6 opérations élémentaires :

- Rotation ω_z d'un quart de la rangée supérieure : c'est la rotation à raison d'un angle droit, dans le sens trigonométrique selon l'axe z, de la face rouge tenue face à soi.
- Rotation ω_z d'un quart de la rangée inférieure : c'est la rotation à raison d'un angle droit, dans le sens trigonométrique selon l'axe z, de la rangée orange tenue face à soi.
- On définit de la même manière les opérations $\omega_x, \omega_x^-, \omega_y, \omega_y^-$.

N'allons pas plus loin dans la formalisation algébrique (mais il n'est pas inutile de consulter la littérature abondante au sujet des groupes d'action, pour approfondir le sujet).

Structure de donnée pour représenter un état

Un tableau 1D suffit

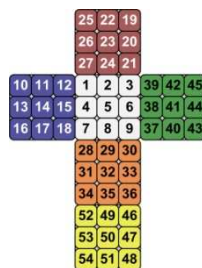
Intéressons-nous en revanche à la conception SDD la plus adaptée pour représenter une position du Rubik's Cube et pour engendrer une partie de l'espace d'état accessible depuis une position donnée, c'est-à-dire pour calculer l'état résultant d'un autre par application de l'une des 6 opérations de base. La théorie des groupes d'actions nous invite à représenter un état comme un n -uplet, et une action comme l'application d'une permutation à ce n -uplet.

En l'occurrence, un état est un 54-uplet avec 9 répétitions de chacune des 6 couleurs, et une action est une permutation du groupe symétrique d'ordre 54.

Poursuivons notre travail de convention en définissant le 54-uplet comme la séquence de 6 blocs des 9 facettes qui occupent respectivement les faces 1 à 6. Pour chacun de ces blocs, considérons les 9 facettes dans l'ordre de parcours en largeur, de haut en bas, de gauche à droite. Précisons les deux axes bas vers haut et gauche vers droite comme les deux axes pris parmi x, y et z , parallèles à la face considérée formant nécessairement un angle d'un quart, tels que, suivant le sens trigonométrique, le premier joue le rôle d'axe gauche vers droite, le second (situé à $\pi/2$ du premier), joue celui d'axe bas vers haut.

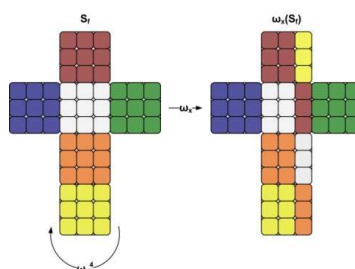
Index

En termes plus intuitifs, et en utilisant pour plan de numérotation des places un patron d'assemblage du Rubik's Cube en état résolu, cela donne ceci :



Ces conventions nous permettent de représenter tous les états licites (ou non) du Rubik's Cube.

Par exemple, pour représenter la rotation ω_x appliquée au Cube depuis son état solution, c'est-à-dire ceci :



L'état du Cube résolu se représente simplement par :

111111112222222233333333444444445555555566666666

En appliquant la rotation ω_x , on atteint l'état :

113113113222222226663333344144144155555555444666666

Plus généralement la rotation ω_x est la permutation des coefficients du vecteur d'état composée de 34 points fixes que l'on peut exprimer sous forme de décomposition en produit de cycles comme :

$(3\ 19\ 46\ 30)(6\ 20\ 47\ 33)(9\ 21\ 48\ 36)(39\ 45\ 43\ 37)(38\ 42\ 44\ 40)$

(Interprétation : c'est la permutation qui amène en position 3 la facette de position 19, en position 19 celle de position 46, en position 46 celle de position 30, en position 30 celle de position 3, ...).

En termes SDD, la conception la plus simple est celle d'un tableau à une dimension de taille 54 qui contient les codes couleurs des facettes. Les 6 opérations élémentaires sont simplement 6 permutations des valeurs du tableau, dont la première vous est donnée ci-dessus.

Pour le reste, à vous de jouer : vous disposez de tous les éléments nécessaires pour démarrer.

OPERATIONS COMPLEXES ET OPTIMISATIONS

Fonction de coût et opérations élémentaires

Sauf à préciser la notion de coût, les 6 opérations de base considérées plus haut vont être associées à un coût unitaire, tandis que des opérations plus complexes vont être associées à un coût qui correspond à leur meilleure décomposition en termes d'opérations de base. Par exemple, une rotation d'un quart dans le sens trigonométrique inverse va être associée à un coût de 3, puisqu'elle peut être obtenue par une séquence de trois rotations d'un quart dans le sens trigonométrique.

Pour ne pas fausser la nature de la fonction de coût, il faut donc compléter les 6 opérations élémentaires de leur opposées respectives. En revanche, cette extension de l'ensemble des opérations « élémentaires » double le nombre d'états à ouvrir lorsqu'un état est visité.

Par le même raisonnement, on peut observer que la rotation d'un quart d'une rangée centrale suivant un axe revient à la rotation de ses deux rangées latérales, d'un quart, dans le sens opposé. Cette combinaison de deux opérations de base est donc également une opération élémentaire et doit être associée à un coût unitaire et être envisagée parmi les transitions qui mènent d'un état à un successeur direct.

Enfin, on peut considérer que les rotations d'un demi-tour, si elles entraînent un effort légèrement supérieur avec un angle double des précédents, n'en demeurent pas moins des opérations élémentaires. Nous déterminons ainsi un ensemble de 27 opérations élémentaires dont la définition et le codage numérique sont donnés dans une table en annexe.

Optimisation algébrique (option pour les plus avancés)

En exploitant la nature algébrique de l'ensemble des opérations sur le cube, on peut envisager d'augmenter la pertinence de l'algorithme au prix d'une consommation mémoire plus importante.

En effet, parmi les combinaisons de deux opérations élémentaires, certaines produisent l'identité tandis que d'autres produisent une opération élémentaire. De même, par les combinaisons de k opérations élémentaires, il s'en trouve qui sont équivalentes à des combinaisons d'ordre inférieur à k .

Pour k fixé, par un travail de réduction algébrique, l'algorithme peut donc constituer une table d'opérations optimisées (leur identification revient à un problème de recherche de plus courts chemins) et leur associer un coût qui est celui du nombre d'opérations élémentaires.

Vous pourrez comparer les coûts algorithmes en termes d'espace et de temps, suivant différents k . et restituer votre analyse en une dizaine de lignes, dans votre fichier **README.txt**.

CAS A RESOUDRE ET TRACE D'EXECUTION

Fournir une trace d'exécution explicite des étapes de votre algorithme notamment des ajouts de nouveaux états dans l'ensemble des ouverts avec évaluation de leurs coût et heuristique, le passage d'un état des ouverts dans les fermés.

Votre programme démarre en chargeant l'état initial du Rubik's à partir d'un fichier nommé **input.txt** et produit un fichier **output.txt**.

Ces fichiers contiennent chacun une séquence d'opérations élémentaires, le premier, celle qui a permis de le « mélanger » depuis l'état résolu, le second, celle qui a permis de le résoudre à l'aide du A*.

Il est impératif de respecter la table de codage des opérations élémentaires fournie en annexe.

Contenu de test suggéré pour **input.txt** :

JHGLDZGLYZ@DPU@IHKGRTJ@AWLCVXZSIASMYKCU@HGAXLMMWDDPNPTEKXR@

REMISE DES TRAVAUX

Livrable

- Un UNIQUE fichier source compilable avec **gcc**, sans nécessiter d'édition de lien.
- Le fichier **input.txt** contenant le cas à résoudre, et le fichier **output.txt** produit par votre programme.
- Un fichier **README.txt** qui comprend impérativement deux sections :
 - Un état d'avancement honnête et fiable de vos travaux en 10 lignes maximum.
 - Les références aux sources qui ont pu inspirer votre conception, la reprise de code étant formellement interdite.
 - Et optionnellement, une mise en avant de fonctionnalités supplémentaires que vous aurez choisi de réaliser dans le but d'obtenir une meilleure note.

Mode de livraison

Le livrable est expédiée comme pièce attachée d'un email dont l'objet est « L'3 GP TP AD 1 NOM1 – NOM2 » adressé à l'animateur de la séance qui vous fournira son adresse en séance.

Une seule remise par binôme.

Délai de livraison

Cinq jours ouvrés après la fin de la séance de TP.

NOTES SUR L'EVALUATION

Comment gagner des points

Critères par ordre décroissant d'importance :

- Le programme compile.
- Le programme s'exécute sans planter.
- Le programme fait ce qu'on attend de lui (partiellement, complètement, avec des plus).
- La trace d'exécution est lisible / intelligible.
- Le code source est efficace (SDD, algorithmes profilés).
- Le code source est propre (indentations, identifiants intelligibles, commentaires bien dosés).

Comment perdre des points

Critères par ordre décroissant de gravité :

- Même code, même partiel, livré par plusieurs équipes : 0 pour les équipes impliquées.
- Retard de livraison : 3 points par jour au prorata temporis.
- Non respect des consignes de packaging du livrable : jusqu'à -3 pts suivant les cumuls.

ANNEXE – TABLE DES OPERATIONS ELEMENTAIRES

Notation	Code	Définition	Commentaire
ω_x	A		Rotation avant d'un quart de la rangée supérieure de l'axe des x
ω_y	B		...
ω_z	C		
ω_x^-	D		
ω_y^-	E		
ω_z^-	F		
$\overline{\omega_x}$	G	ω_x^3	
$\overline{\omega_y}$	H	ω_y^3	
$\overline{\omega_z}$	I	ω_z^3	
$\overline{\omega_x^-}$	J	ω_x^3	
$\overline{\omega_y^-}$	K	ω_y^3	
$\overline{\omega_z^-}$	L	ω_z^3	
ω_{x^2}	M	ω_x^2	
ω_{y^2}	N	ω_y^2	
ω_{z^2}	O	ω_z^2	
$\omega_{x^2}^-$	P	ω_x^2	
$\omega_{y^2}^-$	Q	ω_y^2	
$\omega_{z^2}^-$	R	ω_z^2	
ω_{x^*}	S	$\overline{\omega_x \omega_x^-}$	
ω_{y^*}	T	$\overline{\omega_y \omega_y^-}$	
ω_{z^*}	U	$\overline{\omega_z \omega_z^-}$	
$\overline{\omega_{x^*}}$	V	$\omega_x \overline{\omega_x^-}$	
$\overline{\omega_{y^*}}$	W	$\omega_y \overline{\omega_y^-}$	

$\overline{\omega_z^*}$	X	$\overline{\omega_z \omega_z^*}$	
ω_{x^*2}	Y	$\omega_{x^*}^2$	
ω_{y^*2}	Z	$\omega_{y^*}^2$	
ω_{z^*2}	@	$\omega_{z^*}^2$	