Regular Expression Application in Rubik's Cube Algorithm for Eliminating Redundant Moves

Dhafin Rayhan Ahmad 13518063
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
dhafindiamle@gmail.com

***Abstract*—Over the past few decades, people have been advancing their skills and knowledges to crack further about the twisty Rubik's cube. Different customization and modification have been done to the classic wooden puzzle. Various solving methods are invented in order to break any barrier existing in the world of speedsolving. When algorithmic notations are introduced into the community, the complex sequences applied to the Rubik's cube became more coverable in studies about the cube, not excluding the possible relation between pattern matching and the cube itself.**

Keywords—Rubiks' cube; regular expression; pattern matching; string; algorithm

I. Introduction

For years, people have been passing the time by messing around on a cube-shaped twisty puzzle, the Rubik's cube. Following its popularity in 1980s, many puzzle-enthusiasts raced themselves into beating each other in term of solving time. Some people take Rubik's cube solving into a higher level, such as solving with one-handed, with feet, or even blindfolded. Various methods have been invented in order to solve the famous Rubik's cube. Most of the methods combine the advantages of memorization and intuition.

While a lot of puzzle-solvers are getting faster time in solving the Rubik's cube, some others are interested in finding a shortest possible solution to a scrambled cube. This breakthrough is the one that born a new world in Rubik's cube solving – the fewest move challenge. In this challenge, people are given a pre-generated scramble, and they are required to build up a solution to the scramble, without deriving directly from the scramble sequences. This can be done by using any available solving method, but in order to keep the move count small, fewest move solvers use some more advanced techniques to efficiently minimize moves in their solutions.

II. Rubik's Cube, Fewest Move Solving, and Regex

A. *Rubik's Cube*

The Rubik's cube is a six-sided three-dimensional puzzle, each side usually colored with different colors each other. The cube was invented in 1974 by Hungarian sculptor and professor of architecture Ernő Rubik and get licensed to be sold by Ideal Toy Corporation in 1980. After this agreement, the puzzle start to spread wide around the world, starting the crazy age of solving the Rubik's cube.

The sides of Rubik's cube are recognized with the difference of sticker color stuck on the cube surface, each one of these six colors: white, yellow, green, blue, red, and orange. A popular coloring scheme is to place two similar colors at the opposite side to each other (e.g. red is opposite to orange), although other forms of scheme are also found in the other part of the world, such as the Japanese-scheme in Japan. Some people even use different colors on the cube, using their own color choice in contrast of the six popular one.

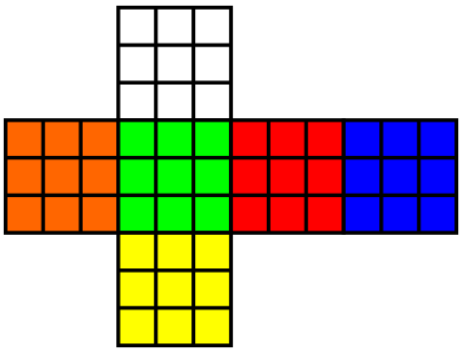

Fig. 1. The standard color scheme for the Rubik's cube.

The Rubik's cube can be seen as a set of faces, which each side has 9 stickers, resulting in a total of 54 faces. A better way to view the cube is to partition the cube into pieces, which is more practical in any solving method. This also gives anyone who learns the cube a better understanding of how the mechanisms work. The puzzle is broken down into three main parts, which is the corners, the edges, and centers. On a normal 3x3x3 Rubik's cube, there are 8 corner pieces, 12 edge pieces, and 6 center pieces. A corner piece can be defined as the piece that has three colors on it, edge piece is where the piece has two colors on it, whereas the center piece has only one color attached to it.

Fig. 2. From left to right, showing the Rubik's cube in highlight of: the corner pieces, the edge pieces, and the center pieces.

A corner piece can only go to the place of the other corners, and so for edges and centers. By this means that, for example, an edge piece cannot go to the place of a corner piece. The positions of center pieces are not changing to each other by any turn, so they are already in their fixed position. Due to this fact, any turns applied to the cube are just actually edges and corners messing around the center pieces.

Some terms are defined for the Rubik's cube to make discussion easier. In a fewest move solving world, it is required to be familiar to the terms of Rubik's cube turning and tracking. Notations are used to make the terms even simpler to understand. These terms will be discussed in a later section.

B. Rubik's Cube Algorithm

In the world of Rubik's cube solving, an algorithm is a set of move sequences that modifies the cube from an initial state to a target state. Algorithms are usually used to solve repeating cases, such as solving specific last layer permutations, cube patterns, scrambles, etc. The main point of algorithms is to give people a communicative and efficient way in sharing their experiences about the Rubik's cube.

Algorithms are used in most popular Rubik's cube solving methods todays. They are usually made to be pre-memorized, and one can recalled the right algorithm to solve the case that he finds while solving the cube in an actual solve. For the most popular intermediate layer-by-layer method, known as the Fridrich's method, algorithms are used in the last two phases out of four total phases available on the method. In advanced fewest moves solving, people use algorithms to share their thought on good commutator moves to solve cases found on fewest moves solving. These algorithms help trigger other solvers to know which commutator they should use to get a more efficient solve during their attempts.

The very basic algorithm involves only six moves from the Rubik's cube, which are turning the six faces of the cube. By using only these six moves, we can get any possible state of the Rubik's cube from any other possible state. In other words, we can cycle through all 43,252,003,274,489,856,000 possible states of the Rubik's cube.

While the basic six moves only apply to the outer face of the cube, there are some variations of algorithm that make it easier for more advanced purpose. These variations include middle slice move, wide move, and rotations. For fewest moves challenge, these advanced moves can help solver for building solution, but many fewest moves solvers still prefer using only the six moves. Restricting the solution to only using the basic moves is also encouraged for better judging purpose.

C. Regular Expression

A regular expression, or regex, is a text string specialized for defining a search pattern. Regular expressions are often treated as wildcards on steroids. For example, for finding all executable files in a file manager, the search query `*.exe` is used. The asterisk character is denoting for the machine to search of any match in replace of itself. This is equivalent to the regular expression `.*\.exe`.

With regular expressions, we can do much more than the wildcard does. A more comprehensive example to demonstrate the possibilities in regular expressions is using the query `\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b` that would match any e-mail address. Regular expressions are used widely in programming. Although, non-programmers also find many utilizations of the regular expressions outside programming. We can use them in powerful search and replace operations to make modifications across large number of files quickly. An easy example would be `colou?r` which will match both spelling of the word color in an operation, instead of two.

There are many text editors that support regular expressions in their search tools nowadays. These are becoming important for developers since developing a project-sized code could be cluttering and they need a way to search for patterns faster. Regular expressions supporting tools sometimes also came up with a cheat sheet for easy understanding of how it operates. The Fig. 3 demonstrates one example of the cheat sheet.

Character classes	
.	any character except newline
\w \d \s	word, digit, whitespace
\W \D \S	not word, digit, whitespace
[abc]	any of a, b, or c
[^abc]	not a, b, or c
[a-g]	character between a & g
Anchors	
^abc$	start / end of the string
\b	word boundary
Escaped characters	
\. * \\	escaped special characters
\t \n \r	tab, linefeed, carriage return
\u00A9	unicode escaped ©
Groups & Lookaround	
(abc)	capture group
\1	backreference to group #1
(?:abc)	non-capturing group
(?=abc)	positive lookahead
(?!abc)	negative lookahead

Quantifiers & Alternation	
a* a+ a?	0 or more, 1 or more, 0 or 1
a{5} a{2,}	exactly five, two or more
a{1,3}	between one & three
a+? a{2,}?	match as few as possible
ab|cd	match ab or cd

Fig. 3. An example of the regular expressions cheat sheet.

III. Notation and Convention

Several notations are used to cover the discussion of turns, permutations, and piece names in the Rubik's cube. Some conventions are also to be introduced to keep the explanation simple.

The Rubik's cube would generally be drawn as a three-dimensional cube showing three sides of it, while the rest are not shown due to perspective angle. Some figures will use translucent cube drawing to show the back sides of the cube. The six sides of the cube are named by the direction their faces are pointing to (up, down, left, right, front, back). The convention is to say the side that appears on the upper part of the figure is up, the one on the left is front, and the other one is right. Therefore, the sides not shown on the figure are down, back, and left, each of them is opposite to up, front, and right, respectively. In the default solved state case shown on this paper, as appears on Fig. 8, the up would be the yellow side, front would be the red side, and the green side for right.

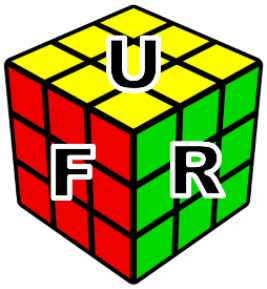

Fig. 4. The three sides of the cube, each letter denoting the initial of their side names.

From here on, we will be using a shorter way to call the six sides, by their initials. For example, U is for up, R is for right, and so on. On a turning sequence, or called an algorithm, moves are written as the six initials, telling which face needs to be turned 90° clockwise. An apostrophe modifier tells us to turn the side 90° counterclockwise (instead of clockwise). Another modifier is to put the number 2 at the end of a letter to denote the 180° turn. Algorithms should be executed in the order as they appear. An example algorithm is $R\ U'\ F2$, which read as "turn the right face 90° clockwise, and then turn the up face 90° counterclockwise, and then turn the front side 180°."

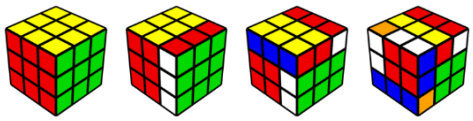

Fig. 5. From left to right: the puzzle on its solved state; R applied; R U' applied; and finally the whole algorithm R U' F2 is applied.

Beside face turns, there are also turns that affect the middle part of the cube called slice turns. The three slice turns are M, S, and E. Slice M is the slice between left and right sides, being turned just as the L move. Slice S is the one between front and back sides, following the turning of F move. The last, E slice is between up and down sides, turns as the D face turns. Modifiers also applies to slice moves. Fig. 6 shows the solved state cube after being applied by the slice moves.

Fig. 6. Appearance of the cube after being applied: the M move, the S move, and the E move; respectively, each from a solved state.

Another important thing to note is the commutators and conjugates notation. A commutator is an algorithm in the form of $A\ B\ A'\ B'$, where either A and B can be a set of turns or just a single turn. Whenever an algorithm meets this condition, it can be written in a shorter notation, $[A, B]$. For example, the commutator $[U'\ R2\ B, L]$ in it's longer form is $U'\ R2\ B\ L\ B'\ R2\ U\ L'$. Notice that the inverse of an algorithm is made by reading the algorithm backward and inversing every individual move on it (180° turn moves mantain the same).

A conjugate is where an algorithm is in the form of A B A'. The form can be written as $[A\colon B]$. We can combine conjugates and commutators on commutator, as in $[D\colon [U\text{^}'\ R'\ U, M']]$, which read as $D\ U'\ R'\ U\ M'\ U'\ R\ U\ M\ D'$.

To build an easier communication, the community of Rubik's cube define a standard guide for naming each sticker place for the cube. Rather than saying "the yellow sticker on the green-yellow-red corner," it is more convenient to say it by the initials of the face associated to the sticker. For example, we would refer the red sticker on the yellow-red-green corner as FRU. The first initial is indicating on which side is the sticker we're meant to, followed by another two sides of the corner, preferably in counterclockwise cycle (saying FUR is still acceptable though). The same goes for the edges, for instance, the yellow sticker on the green-yellow edge is called UR. Please be aware that this labelling system is dependent on the cube orientation we use.

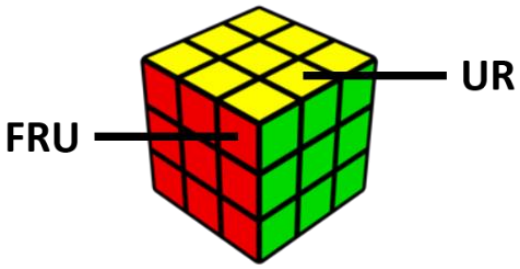

Fig. 7. Example of the labelling system.

Another easy way of naming the stickers is by using letters. For fewest moves solving, it is often encouraged to use this lettering system to make the memorization easier. The letters translate from the normal labelling system as shown on Table 1.

TABLE I. THE LETTERING SYSTEM FOR CORNERS

Sticker Label	Sticker Letter
UBL	A
URB	B
UFR	C
ULF	D
FUL	E
FRU	F
FDR	G
FLD	H
RUF	I
RBU	J
RDB	K
RFD	L
BUR	M
BLU	N
BDL	O
BRD	P
LUB	Q
LFU	R
LDF	S
LBD	T
DFL	U
DRF	V
DBR	W
DLB	X

As seen on Table 1, capital letters are used. To make it different, the edge stickers use noncapital letters instead.

TABLE II. THE LETTERING SYSTEM FOR EDGES

Sticker Label	Sticker Letter
UB	a
UR	b
UF	c
UL	d
FU	e
FR	f
FD	g
FL	h
RU	i
RB	j
RD	k
RF	l
BU	m
BL	n
BD	o
BR	p
LU	q
LF	r
LD	s
LB	t
DF	u
DR	v
DB	w
DL	x

IV. REGEX FOR ELIMINATING REDUNDANT ALGORITHM

The use of regular expressions in pattern searching can be efficiently used to eliminate redundant moves in a given Rubik's cube algorithm. There are several optimization techniques to shorten an algorithm.

A. Direct Repetition

The direct repetition case is the most popular repeating case in Rubik's cube algorithm. This case happens when two or more moves of the same side are used consecutively on an algorithm, regardless of the turning direction. For example, the algorithm $R\ U\ L'L2\ D$ contains two consecutive L moves, which should be eliminated to only one L move. To determine the replacement

move, we should count the total rotation made from the starting of the consecutive moves. For this case, the first L move is 90 degrees counter-clockwise, while the second one is 180 degrees clockwise, so the final result is 90 degrees clockwise. This move is denoted by L, which is obviously shorter than the previous one.

This case is the easiest among the others to identify by sight. But it could be much harder if the algorithm contains long sequences, or if we are checking a lot of algorithms in the same time. This is where the regular expression helps us in finding any redundancy appear on the algorithms. To find consecutive move of the same side, the capturing group is used. The regular expression syntax would be

```
([A-z])[2|']? \1[2|']?
```

where the `\1` is used to refer the first capturing group, which is `([A-z])`. `[A-z]` matches any English alphabet, both lower cases and upper cases. `[2|']?` matches any modifier following a move (can be double-move, reversed, or default).

B. Chain Elimination

The chain elimination case happens when two move can be eliminated from the algorithm, and the leftover sequence takes it to a new redundant case. To illustrate the case, let's an algorithm that consist of chain elimination, $D\ F2\ R'B'B\ R\ F'U\ R2$. The second B move in the algorithms cancels the first B move, leaving the algorithm as $D\ F2\ R'\ R\ F'U\ R2$. Now, we can see that the R moves also cancelling themselves. We now have $D\ F2\ F'U\ R2$. From here, the two F moves form the direct repetition case which could be translated to just F.

The regular expressions can handle the case for limited number of moves; for example, for two chain of cancelling moves we can use

```
([A-z])[2|']? ([A-z])[2|']? \2[2|']? \1[2|']?
```

which modifies the basic syntax in direct repetition to allow chained case by using multiple capturing group.

C. Interspersed Repetition

The interspersed repetition is a redundant case that happens when two direct repetition is interspersed by a move of the opposite side. For example, the algorithm $U\ F\ R2\ L\ R\ D$ have two R moves interspersed by an L move. The two moves can be eliminated and replaced by the appropriate replacement move before or after the interspersing move.

The regular expressions for this case should be more specific to the move, for example

```
R[2|']? L[2|']? R[2|']?
```

which handle the case for when two R moves are interspersed by an L move.

D. Complex Notation

Sometimes, the redundant moves in an algorithm is hard to be spotted because of the use of a more advanced notations, such as the notations that denote rotations, wide moves, slice moves, etc. This requires a sharper view and experience to identify the redundant part by sight, or one can simply apply the algorithm to a real cube to see if there any one of the sequences that is redundant.

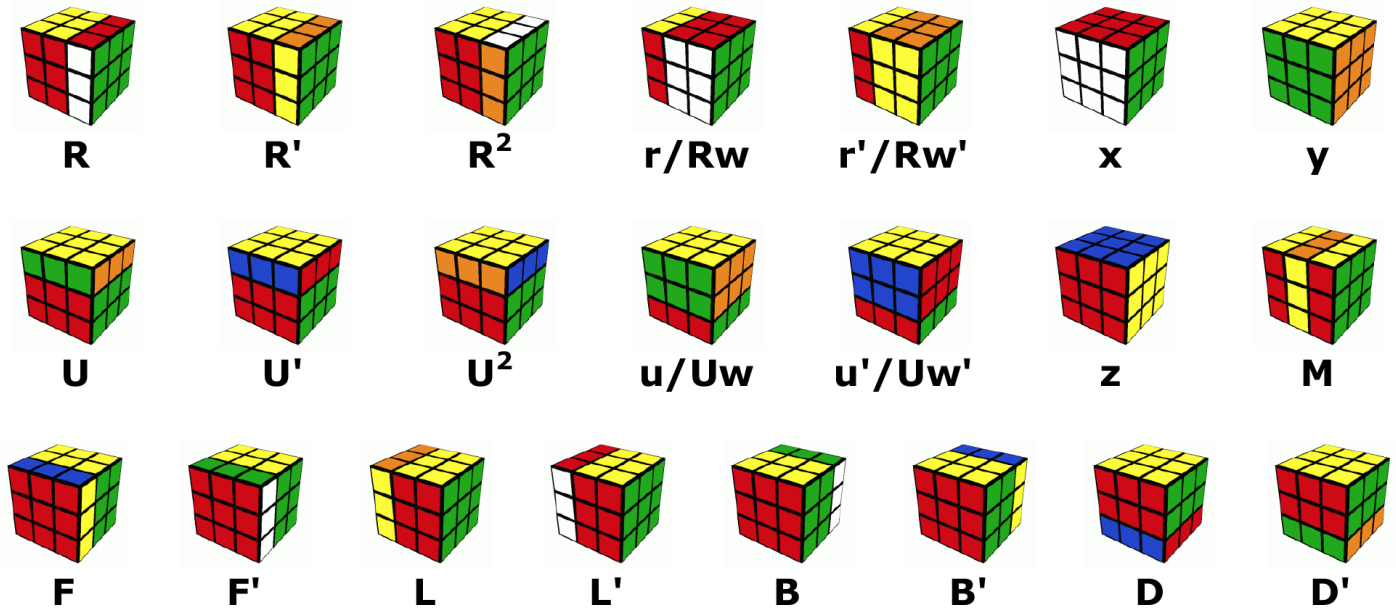

Fig. 8. The complete guide to Rubik's cube notations.
(*Source: https://medium.com/@maxdeutsch/m2m-day-69-decoding-rubiks-cube-algorithms-6ea7e7704ec9)*

APPENDIX

All Rubik's cube models shown in this paper are generated from an open source visual cube generator from http://cube.crider.co.uk/visualcube.php.

VIDEO LINK AT YOUTUBE

A video for explaining the application of regular expressions in eliminating redundant algorithm is also available on YouTube, which can be accessed in https://youtu.be/B8o_s-EFcXM.

ACKNOWLEDGMENT

All praise to Allah, only with His guidance I could finish this paper. After that, I thank Dr. Ir. Rinaldi Munir, M.T. for his guide in understanding the algorithm strategies in the class. Special thanks to Ernő Rubik for his amazing invention on the cube. I also appreciate to the great community of cubing, especially Dan Haris, for making a breakthrough start in the world of fewest moves Rubik's cube solving.

REFERENCES

[1] R. Tran, "A Mathematical Approach to Solving Rubik's Cube" UBC Math38. Fall 2005.

[2] J. Goyvaerts, Regular Expressions: The Complete Tutorial, 2007.

[3] D. Rayhan, "Graph Application in Rubik's Cube for Blindfolded Solving". 2019.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 4 Mei 2020

Dhafin Rayhan Ahmad 13518063