Chengyue (Melissa) Wu 1930284

420-LCU Computer Programming, Section 3

S. Hilal, instructor

External Documentation for Final Project

# CubeSolver

## Part A - User Manual

The program is able to solve a 3 by 3 Rubik's Cube using CFOP. The targeted users include but not limited to (1) people who do not know how to solve a 3 by 3 cube; (2) people who just started to learn to solve a 3 by 3 cube, ie. beginners; (3) people who are learning or would like to learn the method CFOP; (4) anyone can use it for fun. To use this program, you need to enter a scramble (a sequence of basic moves) at any length, the program will then return a solution algorithm, which is not the inverse of your scramble.

## Part B - Internal Workings Explained

The program bottoms on a mathematical basis. Concepts in linear algebra such as vectors and linear transformation are used to mathematically determine the cube itself and permutations.

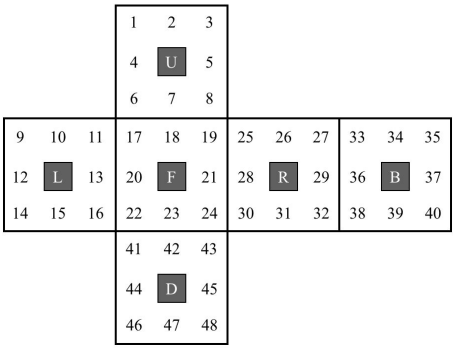


*Image 1: flat representation of numbered initial cube*[1]

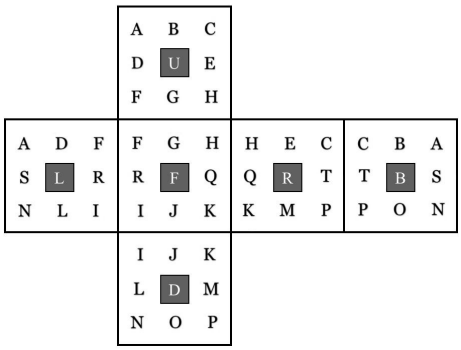


*Image 2: flat representation of initial cube with vectors*

Here's how we translate a Rubik's cube into numbers: First, we number each facet[2] of the cube (excluding centers) 1-48 as shown in the Image 1. Then we place the cube in a 3-D Euclidean Space, choosing the center of the cube as the origin. The twenty six individual pieces of the cube are then represented by twenty six different vectors in R3, from vector a = (-1, -1, 1) to vector z = (0, 0, -1), as shown in Image 2. In this way, we are able to mathematically

[1] http://www.sfu.ca/~jtmulhol/math302/puzzles-rc-representation.html

[2] facet: the individual stickers on the cube

determine an unscrambled cube: for example, the top right corner of the cube (capital A as shown in Image 2) is a: [35, 9, 1] where a = (-1, -1, 1). The order of the three numbers in the square bracket is determined by the geometric relationship between the corresponding facets and the three axes. For example, the facets labeled "35" is orthogonal to the x-axis, the one labeled "9" is orthogonal to the y-axis and the one labeled "1" is orthogonal to the z-axis, as it follows the order [x, y, z], it is therefore written as [35, 9, 1]. When it comes to an edge piece who only has two facets, we use "0" to replace the missing facet. For example, b is [34, 0, 2]. Centers are represented in a similar way but with a string directly indicating the face[3] instead of numbers: center of the front layer is w: ['F', 0, 0].

Now that the initial cube is translated into pure math, the next step is to use linear transformation to perform basic permutations. Basic permutations are U (upper layer rotates 90 degree clockwise), its inverse U' (upper layer rotates 90 degree counter-clockwise), F and F', R and R', D and D', B and B', L and L', M and M' (there are also E and E', S and S', but they are not used in this program). We know that rotation in R3 is a linear transformation since we can multiply the original vector in R3 by a 3 by 3 matrix to get a new vector still in R3: $T: R3 \rightarrow R3$, $T(v) = Av$. In this way, we are able to find the new position of a piece after a permutation. Using the piece a:[35, 9, 1] as an example, we perform U on the cube, that is, a 90 degree rotation about the z-axis clockwise, which can be realized by multiplying the 3 by 3 matrix [[0, 1, 0], [-1, 0, 0], [0, 0, 1]] which we will call it "Tz". In mathematical terms, the linear transformation can be written as U(a) = Tz*a = [[0, 1, 0], [-1, 0, 0], [0, 0, 1]] * (-1, -1, 1) = (-1, 1, 1) = c. Therefore c = (-1, 1, 1) is the new position for the piece [35, 9, 1] as expected if a U move is done on an actual cube. However, notice that after the piece goes to c, the facet orthogonal to x-axis is now labeled "9" instead of the original "35", and the facet orthogonal to y-axis is now labeled "35" instead of the original "9", while z remains the same which is still "1". We conclude that when a 90-degree rotation about the z-axis, either clockwise or counter-clockwise, is performed, facet x and y swap with each other, while facet z stays unchanged. Similarly, in a 90-degree rotation about the x-axis, y, z swaps and x stays; in a 90-degree rotation about the y-axis, x, z swaps and y stays.

[3] face/Layer: the nine-piece 3D face

In the program, I created a dictionary to represent the cube, using the twenty six vectors as keys and numbered facets as values. Everytime a basic permutation is executed, three things are done on the code: firstly, calculate the new position of the piece; secondly, give its new orientation; finally, the value of the new key/vector is then changed to that piece with modified order of the three numbers representing the three facets. Each algorithm or sequence of moves is a combination of basic permutations, therefore the cube dictionary can keep track of any moves done on the cube.

After the basic settings are done, we just need to analyze the cube dictionary and apply CFOP (Cross, First+Second Layer, 2-Look OLL, PLL) to it. The key is to determine where the targeted facets are and where we want them to be, by comparing them with an unscrambled reference cube dictionary. I use various mathematical tools from arithmetic -- in order to track the relative position of two or more facets, to permutation and combination to list all the possible cases with the help of if statement and while and for loop. Finally, the program takes a scramble at any length as the user input and returns a simplified solution algorithm with no extra moves, such as U followed by U’ which can be canceled out, by using list operations.