Rubik’s Cube Solver

Trent Rand, Robert Dinaro

Abstract— **Solving a Rubik’s Cube is no easy task. Typically, a user must follow an algorithm, performing dozens of calculated rotations on different sides of the cube to solve it. Through the method of computing permutations of side rotations on a cube map, any Rubik’s Cube can be solved. The objective of this research is to prototype a method for brute forcing the solution for a Rubik’s Cube - thus creating a Rubik’s Cube solver. While a brute force approach is not the most ideal method for computing a solution, it does provide accurate results in a timely manner. Through this research, our prototype can solve any Rubik’s Cube configuration, if it is valid; however, further enhancements must be done to decrease the computational cost of solving a complexly scrambled Rubik’s Cube.**

I. INTRODUCTION

The Rubik’s Cube is a logic puzzle toy invented in 1974 by Hungarian sculptor and professor Ernö Rubik. Rubik’s invention was first called the *Magic Cube*; however, in 1980 the toy was licensed to Ideal Toy Corporation, where it was therefore advertised as the *Rubik’s Cube*. Since its date of creation, the Rubik’s Cube has sold over 350 million cubes - a number which declares the Rubik’s Cube as the world’s top-selling puzzle game.

The Rubik’s Cube is a six sides puzzle, each of which having the ability to rotate freely in 90 degree intervals. Every side is composed of nine smaller cubes, each of which being assigned a specific color - typically among the color set of white, red, blue, orange, green, and yellow. To solve the puzzle, each face must be returned to a state where only one color is present per side - thus having nine cubes of a particular color per side. The middle cube on each side is immutable, meaning its position cannot be moved. For this reason, the color of the eight surrounding cubes on each side must be matched to the color of the middle cube in order to solve a side. When all sides have been solves, the Rubik’s Cube is considered solved.

A typical process for solving the Rubik’s Cube involves a series of algorithms which each accomplish specific tasks. For example, a user should first attempt to solve the edges on the top side of the cube. The top side of the cube can be considered any side; however, once decided upon it must remain as that for the duration of the solution. The edges are the innermost pieces which are directly perpendicular with the static middle cube. Once the edges are solved, the user then solves the edges of the top side, which are the four outside-most pieces. The user would then move on to solving the middle layer, then the top sides, then the top corners. At this point, after performing several algorithms for solving each portion previously mentioned, the user should then have a solved Rubik’s Cube.

II. PROPOSED METHOD

Because there are over 43 quintillion permutations, we wanted to focus on the idea of brute forcing a solution to the Rubik’s Cube using only 6 distinct operations. To achieve this, we needed two key components - a model in which to represent the Rubik’s Cube in code, and a set of rules that mimic the operation of rotating a side on the cube by 90 degrees clockwise.

To represent the Rubik’s Cube in code, we created a structured data model which we refer to as a *cube map*. The documentation of our project features a well-defined set of guidelines for the user to follow when entering their cube into the program. To further elaborate, it is worth formerly defining some basic properties of the Rubik’s Cube:

- Side: Any of the 6 faces of the Rubik’s Cube, each being defined by the color of the static middle cube.
- Color: The color of a particular cube, each of which being represented by the first letter of their name (e.g. White is referred to as *w*, Red is referred to as *r*).
- Rotation: The action of rotating a particular side by 90 degrees clockwise; where the rotation of clockwise is from the perspective of the user holding the cube (i.e. if rotating the front or the back, both would spin in the same clockwise direction).
- Orientation: The orientation in which the user is referring to the Rubik’s Cube. No particular orientation must be used; however, once deciding on once, that orientation must remain consistent throughout the processes. A suggested orientation is with the white side on the top and red side facing the user.
- Solved: The Rubik’s Cube is considered solved when all the six sides contain only one color each.

The cube map data structure consists of 54 colors, each of which representing the color of a particular cube. The cube map follows a well-defined order, starting with the top left color of each side and moving left-to-right row-by-row until reaching the bottom right color. This order remains consistent for each side. The order of entering the sides is as follows - Up, Down, Left, Right, Front, Back.

EXAMPLE OF A SOLVED CUBE MAP

```
cube(w, w, w, w, w, w, w, w, w, y, y, y, y, y, y, y, y, y, g, g, g, g, g, g,
g, g, g, b, b, b, b, b, b, b, b, b, r, r, r, r, r, r, r, r, r, o, o, o, o, o, o, o, o,
o)
```

With the well-defined cube map data structure, half of the problem is solved. The remaining factor involves devising a set of operations which mimic the action of a user rotating any of the six sides. This was done by defining generic mappings which translate a given initial state of the cube map into the representative translation after applying an operating of rotating the chosen side by 90 degrees clockwise.

To build these *rotation* translation maps for each of the six clockwise side rotations, we defined cube maps which consist entirely of 54 variables; each of which representing the current color of that particular cube (i.e. the current color that exists in the particular index of the cube map). Each side rotation has a *before* and *after* cube map model, which maps a cube's color from a particular index in the *before* cube map to its new position in the *after* cube map after applying the chosen side rotation operation.

EXAMPLE OF A TOP SIDE ROTATION

Before:

```
cube(
    W1, W2, W3, W4, W5, W6, W7, W8, W9,
    Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9,
    G1, G2, G3, G4, G5, G6, G7, G8, G9,
    B1, B2, B3, B4, B5, B6, B7, B8, B9,
    R1, R2, R3, R4, R5, R6, R7, R8, R9,
    O1, O2, O3, O4, O5, O6, O7, O8, O9
)
```

After:

```
cube(
    W7, W4, W1, W8, W5, W2, W9, W6, W3,
    Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9,
    G1, G2, R1, G4, G5, R2, G7, G8, R3,
    O7, B2, B3, O8, B5, B6, O9, B8, B9,
    B7, B4, B1, R4, R5, R6, R7, R8, R9,
    O1, O2, O3, O4, O5, O6, G9, G6, G3
)
```

This operation may be hard to comprehend without a physical representation. If you'd like to further understand how this works, try labeling a solved Rubik's Cube with the letter/number combination referring to each pieces color and position (e.g. the white side should be labeled W1 - W9 in the order previously specified).

Now that a well-structured cube map data structure has been defined alongside operations that can be applied on the cube map to mimic manipulation of the cube, computing a brute force solution to the Rubik's Cube is simple. A base case of the solved cube is created as a cube map, as previously shown. By computing all permutations of applying Rotation translations maps to the user-entered cube, a solution can be found when the cube map returned after a translation is applied matches the solved cube map base case. Along the way, each *branch* of permutations keeps track of its previously applied operations. When the solution is found, these applied operations are returned to the user, allowing the user to apply the operations to their physical Rubik's Cube to solve it.

EXAMPLE OF RUNNING THE RUBIK'S CUBE SOLVER

```
| ?-  solve(Solution, cube(g, w, w, g, w, w, r, r, r, b, b, o, y, y, o, y, y,
o, g, g, r, g, g, r, y, y, b, o, b, b, o, b, b, g, b, b, w, w, w, r, r, y, r, r, y,
o, o, w, o, o, w, y, g, g), C), solved(C).

Solution = [up, up, up, right, right, right, front, front, front]
```

COMPARISON OF COMPUTATIONAL COST BY COMPLEXITY

Operations applied to a solved Rubik's Cube	**Time required to find a solution**	**Operations required to solve**
1 operations	19 ms	3
2 operations	40 ms	6
3 operations	8126 ms	9
4 operations	1749068 ms	12

Fig. 1. Computational cost required to solve a Rubik's Cube with a variable number of operations initially applied from a solved state

III. DISCUSSION

One of the difficulties we faced while solving the Rubik's Cube was the poor performance in computational time for solving the cube. In figure 1, when three operations were applied to the cube, the time required to find a solution was 8,126 milliseconds, then when four operations were applied, the time increased a substantial amount to 1,749,068 milliseconds. Each extra rotation applied to the solved cube increased the time required to find a solution exponentially. This was a result of our solution having no counterclockwise rotations, causing the computational time for solving to increase exponentially with each additional step required. A proposed solution to optimize the counter-clockwise rotation scheme is to stope occurrences where if a clockwise rotation occurs, there should not be a counterclockwise rotation in the same direction.

References

During the research and development of our Rubik's Cube solver, we used the following resources to discover and analyze methods used in existing approaches, as well as to prepare statistical and factual information:

[1] https://en.wikipedia.org/wiki/Rubik%27s_Cube

[2] https://www.rubiks.com/solve-it

[3] https://github.com/trentrand/rubikssolver

Appendix 1

A condensed but complete version of the source code for our solution has been included. The complete project can be found in our code repository[3] on GitHub.

rubiks.pl

```
# Author: Trent Rand, Robert Dinaro
# Date: April 28, 2017
% Define the base case, where all colors are in correct position and the cube is solved
solved(cube(W,W,W,W,W,W,W,W,W,Y,Y,Y,Y,Y,Y,Y,Y,Y,G,G,G,G,G,G,G,G,G,B,B,B,B,B,B,B,B,B,R,R,R,R,R,R,R,R,R,O,O,O,O,O,O,O,O,O)).


% Solve the Rubik's Cube by inputting your cube map as variable Cube
solve([], Cube, Cube).
solve([NextRotation | Rotation], Cube, NewState) :- solve(Rotation, CurrentState, NewState), rotateside(NextRotation, Cube, CurrentState).

% Side Rotation translation maps-


% Rotate the Up side clockwise by 90 degrees
rotateside(
   up,
   % translate cube map from:
   cube(
      W1, W2, W3, W4, W5, W6, W7, W8, W9,
      Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9,
      G1, G2, G3, G4, G5, G6, G7, G8, G9,
      B1, B2, B3, B4, B5, B6, B7, B8, B9,
      R1, R2, R3, R4, R5, R6, R7, R8, R9,
      O1, O2, O3, O4, O5, O6, O7, O8, O9
   ),
   % to new cube map with Up side rotated clockwise by 90 degrees:
   cube(
      W7, W4, W1, W8, W5, W2, W9, W6, W3,
      Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9,
      G1, G2, R1, G4, G5, R2, G7, G8, R3,
      O7, B2, B3, O8, B5, B6, O9, B8, B9,
      B7, B4, B1, R4, R5, R6, R7, R8, R9,
      O1, O2, O3, O4, O5, O6, G9, G6, G3
   )
).


% Rotate the Down side clockwise by 90 degrees
rotateside(
   down,
   % translate cube map from:
   cube(
      W1, W2, W3, W4, W5, W6, W7, W8, W9,
      Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9,
      G1, G2, G3, G4, G5, G6, G7, G8, G9,
      B1, B2, B3, B4, B5, B6, B7, B8, B9,
      R1, R2, R3, R4, R5, R6, R7, R8, R9,
      O1, O2, O3, O4, O5, O6, O7, O8, O9
   ),
   % to new cube map with Down side rotated clockwise by 90 degrees:
   cube(
      W1, W2, W3, W4, W5, W6, W7, W8, W9,
      Y3, Y6, Y9, Y2, Y5, Y8, Y1, Y4, Y7,
      R7, G2, G3, R8, G5, G6, R9, G8, G9,
      B1, B2, O1, B4, B5, O2, B7, B8, O3,
      R1, R2, R3, R4, R5, R6, B9, B6, B3,
      G7, G4, G1, O4, O5, O6, O7, O8, O9
   )
).


% Rotate the Left side clockwise by 90 degrees
rotateside(
   left,
   % translate cube map from:
   cube(
      W1, W2, W3, W4, W5, W6, W7, W8, W9,
      Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9,
      G7, G4, G1, G8, G5, G2, G9, G6, G3,
      B1, B2, B3, B4, B5, B6, B7, B8, B9,
      R1, R2, R3, R4, R5, R6, R7, R8, R9,
      O1, O2, O3, O4, O5, O6, O7, O8, O9
   ),
   % to new cube map with Left side rotated clockwise by 90 degrees:
   cube(
      O1, W2, W3, O4, W5, W6, O7, W8, W9,
      R1, Y2, Y3, R4, Y5, Y6, R7, Y8, Y9,
      G7, G4, G1, G8, G5, G2, G9, G6, G3,
      B1, B2, B3, B4, B5, B6, B7, B8, B9,
      W1, R2, R3, W4, R5, R6, W7, R8, R9,
      Y1, O2, O3, Y4, O5, O6, Y7, O8, O9
   )
).


% Rotate the Right side clockwise by 90 degrees
rotateside(
   right,
   % translate cube map from:
   cube(
      W1, W2, W3, W4, W5, W6, W7, W8, W9,
      Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9,
      G1, G2, G3, G4, G5, G6, G7, G8, G9,
      B1, B2, B3, B4, B5, B6, B7, B8, B9,
      R1, R2, R3, R4, R5, R6, R7, R8, R9,
      O1, O2, O3, O4, O5, O6, O7, O8, O9
   ),
    % to new cube map with Right side rotated clockwise by 90 degrees:
    cube(
        W1, W2, R3, W4, W5, R6, W7, W8, R9,
        Y1, Y2, O3, Y4, Y5, O6, Y7, Y8, O9,
        G1, G2, G3, G4, G5, G6, G7, G8, G9,
        B7, B4, B1, B8, B5, B2, B9, B6, B3,
        R1, R2, Y3, R4, R5, Y6, R7, R8, Y9,
        O1, O2, W3, O4, O5, W6, O7, O8, W9
    )
).


% Rotate the Front side clockwise by 90 degrees
rotateside(
    front,
    % translate cube map from:
    cube(
        W1, W2, W3, W4, W5, W6, W7, W8, W9,
        Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9,
        G1, G2, G3, G4, G5, G6, G7, G8, G9,
        B1, B2, B3, B4, B5, B6, B7, B8, B9,
        R1, R2, R3, R4, R5, R6, R7, R8, R9,
        O1, O2, O3, O4, O5, O6, O7, O8, O9
    ),
    % to new cube map with Front side rotated clockwise by 90 degrees:
    cube(
        W1, W2, W3, W4, W5, W6, G7, G8, G9,
        B9, B8, B7, Y4, Y5, Y6, Y7, Y8, Y9,
        G1, G2, G3, G4, G5, G6, Y3, Y2, Y1,
        B1, B2, B3, B4, B5, B6, W7, W8, W9,
        R7, R4, R1, R8, R5, R2, R9, R6, R3,
        O1, O2, O3, O4, O5, O6, O7, O8, O9
    )
).


% Rotate the Back side clockwise by 90 degrees
rotateside(
    back,
    % translate cube map from:
    cube(
        W1, W2, W3, W4, W5, W6, W7, W8, W9,
        Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9,
        G1, G2, G3, G4, G5, G6, G7, G8, G9,
        B1, B2, B3, B4, B5, B6, B7, B8, B9,
        R1, R2, R3, R4, R5, R6, R7, R8, R9,
        O1, O2, O3, O4, O5, O6, O7, O8, O9
    ),
    % to new cube map with Back side rotated clockwise by 90 degrees:
    cube(
        G1, G2, G3, W4, W5, W6, W7, W8, W9,
        Y1, Y2, Y3, Y4, Y5, Y6, B3, B2, B1,
        Y9, Y8, Y7, G4, G5, G6, G7, G8, G9,
        W1, W2, W3, B4, B5, B6, B7, B8, B9,
        R1, R2, R3, R4, R5, R6, R7, R8, R9,
        O3, O6, O9, O2, O5, O8, O1, O4, O7
    )
).
```