Group 2: Rubik's Cube Project Report

Project Summary

This project is based on using an algorithm in Python to solve a 3x3 randomized Rubik's Cube. The algorithm is divided into six separate stages and each stage solves either a face or row of the cube before moving on to the next stage. We then converted our algorithm from Python to MATLAB to take advantage of the native Graphical User Interface in MATLAB. In this GUI, the user enters the colors for each piece of the cube and the algorithm solves the cube and gives the user the list of steps to solve the cube for themselves.

Section 1: Data Structures / Algorithms

There were several different data structures involved in our algorithm used for the cube in memory, the list of moves to replicate the steps, and a variety for the GUI.

The Rubik's Cube algorithm is based on using a set of moves that are predefined based on the fact that there are only so many possible moves one can apply to the cube. Therefore, we defined these moves in which the columns or rows of the phases of the cube change. These moves then rearrange the respective contents of the 2D representation of our cube depending on which moves were performed.

We first created a two-dimensional array that stores the whole cube where each entry is the starting character of each color. This two-dimensional array is the foundation of our algorithm because this is the representation of the cube. After each move is applied to the cube while solving, the corresponding move is added to the list of moves in order to give the user the steps we used. Once the algorithm is finished, the entries of the array should contain a solved cube and the list will give the exact steps to solve the cube. In order to represent the cube in MATLAB, we implemented a cell data structure which is a matrix of strings used to color each piece of the cube in our GUI. Finally, each element of the GUI is stored as an object. We can then manipulate each object to store specific entries. For example, there is an object that stores the string of the next move, and we change that string after each move.

Section 2: Complexity Analysis / Optimization

We found the time complexity of the entire algorithm is O(n^2). To do this, we found the time complexity of each individual stage which is outlined below.

For the first stage, assuming it is a 3x3x3 cube, the algorithm would have to check at most every side which is 54 spots. This is obtained by 6 sides * the number of faces on each side which is n^2. The number of sides that would have to be checked for any arbitrary cube would be 6 * n^2. The asymptotic analysis of this would then be O(n^2).

For the second stage, if we take a 3x3x3 cube, the algorithm would check all the corner pieces of all the faces (except the bottom face) to make a swap, so that makes it 5 * 4 * (n-2) = 20 where n = 3. However, for a general case, it becomes 20(n-2) where n is the number of pieces on a side. So the complexity is O(n).

The next stage is similar to the second stage per row. To get a single row, this would be 16n. However, this process would have to be then repeated (n-2) times to get the sides. This would then come out to be O(n^2) in an asymptotic analysis.

The fourth stage, which is solving the top face, is similar to the first face except now almost all of the sides are complete. This complexity would come out to be O(n^2).

The time complexity for the final stage of the Rubik's cube is on average O(n) since we are only checking the top face of the cube and the worst case scenario would be that it is checked 4n times.

The time complexity of the optimization is O(n) for all cases, where n is the number of moves after all the previous stages are compiled together and the two patterns are O(1) for all cases.

Because the largest component is O(n^2), the time complexity of the entire algorithm is O(n^2).

In order to solve the Rubik's cube, it is expected that it would require a large number of moves, some of which may be redundant. Each rotation of the rubik's cube is complemented to another rotation, for example, a "B" rotation has the complement "Bi", "R" has the complement "Ri", and vice versa. To optimize the process of solving the cube, the redundant steps of three same consecutive rotation is replaced with its complement. This method has significantly reduced the numbers of steps to take in solving the cube.

In addition, the average case scenario would be the worst case scenario. The probability that there is not at least one color on each side to start is extraordinarily small. In a probabilistic model, each piece is independent of every other piece on that side. Due to independence, we can then apply some simple probabilistic counting. The probability that an arbitrary face is solved is (⅙)^((n^2)-1). On a 3x3x3 cube, this probability is 5.9e-7 which is extremely small. As the cube gets larger, the probability decreases significantly. For example, a 5x5x5 cube would have the probability of a face being solved is 3.23e-19. As the cube gets large, the probability goes way down. We can therefore conclude that we cannot assume that a cube face will be partially solved and this proves that the average case scenario is also the worst case scenario.

Section 3: Result and Testing Procedure

The full extent of the program ultimately gives the end user instructions to solve a scrambled 3x3 rubik's cube. In addition, the GUI created in MATLAB gives a visual representation along with

the list of steps so that the user can follow along. The user also has the option to transform their already solved rubik's cube into two patterns: "Multicolored Cross" and "Square in the Middle".

Throughout the whole project, an algorithm to generate a randomized rubik's cube allowed us to test the logic of each stage. There are specific constraints to a rubik's cube such as having nine of each color, specific corner and edge pieces. As any good programmer knows, you should attempt to break your own code. To do that, we ran about 140,000 test cases and every single one worked. To time this algorithm, we tested 5000 (scrambled) rubik's cubes and from start to finish it took an average of 2 minutes. This implies that a each rubik's cube was solved in less than 30 milliseconds.

Section 4: Work Distribution

Michael

- Team Leader
- The individual moves (Code Framework)
- First Stage of solution
- Integrated all the stages together
- Converted all the code to MATLAB
- Created GUI and backend code for GUI
- Error checking for GUI

Ruchir

- Stage 2: Solving the corner pieces of the bottom face and the bottom layer

Apoorv

- Stage 3: Solving the middle layer

Arjun

- Stage 4: Solving the Top Face, using a 2 Step Method (Cross first, then full face)

Imad:

- Stage 5: Algorithm to position Yellow Corners and Edges correctly in order to get a solved Rubik's Cube.

Gabriel:

- Step 6: Optimization, reducing redundant steps.
- Function for "Multi-colored Cross" and "Square in the Middle".

Citation (APA):

Description: This website describes the algorithm that is used to solve the rubik's cube. We converted the basic logic given in this guide into code.

1. Mao, T. & Lee, J. (2014). *HOW TO SOLVE THE RUBIK'S CUBE*. *Rubiks*. Retrieved 28 April 2016, from https://rubiks.com/blog/how-to-solve-the-rubiks-cube

Description: The following citations are for the images used in the presentation

2. Some Rubik's Cube Art... (2006, May 24). Retrieved April 28, from https://www.speedsolving.com/forum/threads/some-rubiks-cube-art.250/
3. How to solve the white corners in the first layer of the Rubik's Cube. (n.d.). Retrieved April 27, 2016, from http://ruwix.com/the-rubiks-cube/how-to-solve-the-rubiks-cube-beginners-method/step-2-first-layer-corners/
4. *I0.wp.com*. Retrieved 27 April 2016, from http://i0.wp.com/www.subtwentycubing.com/wp-content/uploads/2014/11/First-Layer-Corners.png
5. *How to solve a Rubik's Cube | The ultimate beginner's guide*. *Singapore champion guide on How to solve a Rubik's Cube*. Retrieved 28 April 2016, from http://www.learnhowtosolvearubikscube.com/how-to-solve-a-rubiks-cube-solution-overview/
6. *SolveTheCube*. *Solvethecube.com*. Retrieved 28 April 2016, from http://solvethecube.com

7. HOW TO SOLVE THE RUBIK'S CUBE - STAGE 6. (2014, June 19). Retrieved April 28, 2016, from https://rubiks.com/blog/how-to-solve-the-rubiks-cube-stage-6