A) **PROJECT Type**

A “real world” **APPLICATION** of mathematics

B) **TARGET AUDIENCE**

Our project is based on Rubik's cube which is considered a permutation group by group theory. With these program will can discover the average number of random moves it takes for 2 or 3 selected points to get to the same side. We will also see how the rubik’s cube is a group by testing different moves (element) and computing that set of moves until it comes back to its original state.

C) **PURPOSE AND BACKGROUND**

The Rubik’s Cube made up of 6 faces, each with 8 rotating squares around the middle piece. Making it a quintessential example of group theory puzzle. It was invented in 1974 by the Hungarian Erro Rubik puzzle. By 2009 more than 350 million Rubik’s Cubes had been sold. The current record time for solving it is under 3.475 seconds: under 17 seconds blindfolded. Although it was proved in 1995 that there was a starting configuration that required at least 20 moves to solve, it was not until 2010 that it was determined that every cube could be solved in at most 20 moves.

We can make the set of moves of the Rubik’s cube into a group, which we will denote $(G, *)$. The elements of G will be all possible moves of the Rubik’s cube (for example, one possible move is a clockwise turn of the top face followed by a counterclockwise turn of the right face). Two moves will be considered the same if they result in the same configuration of the cube (for instance, twisting a face clockwise by 180° is the same as twisting the same face counterclockwise by 180°). The group operation will be defined like this: if M1 and M2 are two moves, then $M1 * M2$ is the move where you first do M1 and then do M2.

Why is this a group? We just need to show the 4 properties

- G is certainly closed under $*$ since, if M1 and M2 are moves, $M1 * M2$ is a move as well.
- If we let e be the “empty” move (that is, a move which does not change the configuration of the Rubik’s cube at all), then $M * e$ means “first do M, then do nothing.” This is certainly the same as just doing M, so $M * e = M$. So, $(G, *)$ has a right identity.
- If M is a move, we can reverse the steps of the move to get a move M0 . Then, the move $M * M0$ means “first do M, then reverse all the steps of M.” This is the same as doing nothing, so $M * M0 == e$, so M0 is the inverse of M. Therefore, every element of G has a right inverse.

- Finally, we must show that $*$ is associative. Remember that a move can be defined by the change in configuration it causes. In particular, a move is determined by the position and orientation it puts each cube in.

D) **CHECKLIST OF FEATURES**

Program 1: - Average number of random moves until 2 or 3 selected points get to the same side.

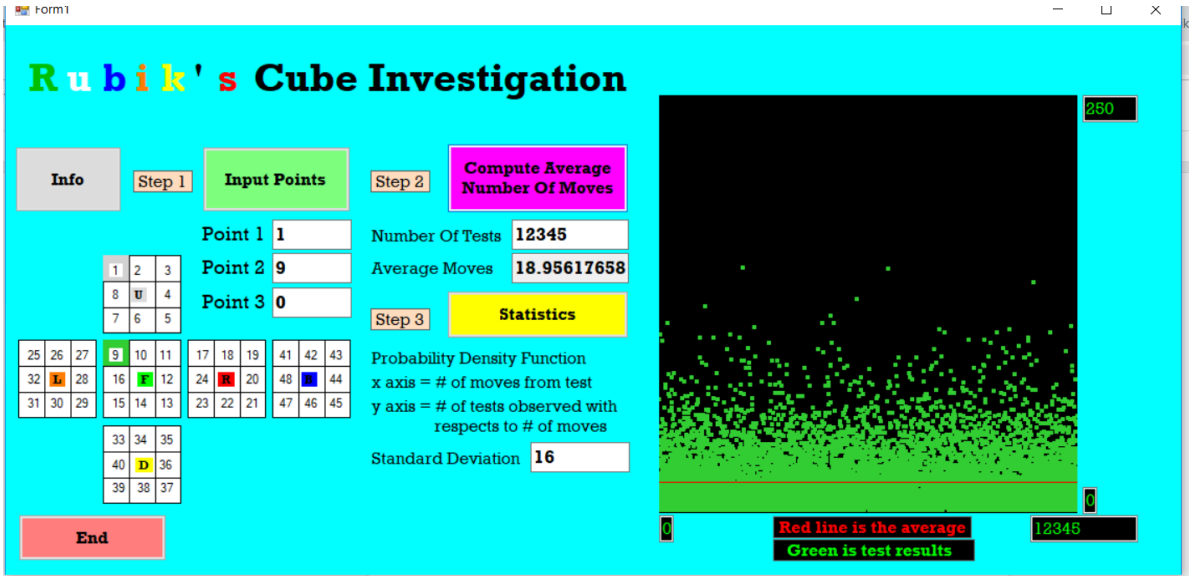
- Plots all tests on a graphs and displays an average line.
- Plots a probability density function with the results. Also finds Standard Deviation.
- Display that shows choices of points to pick by the user and will highlight the choices once they are entered in 6 different pictureboxes to represent each face of the cube.
- Info button brings user to a second form that describes the overall program and how to use it.

Program 2: - Computes moves entered into program by user as an element. Displays the total number of sequences used to get the cube back to original state.

- Handwritten examples of our group theory observations of the rubik's cube. Examples of complete computations using group theory. Program will confirm.

E) **SUMMARY OF OBSERVATIONS**

Project 1: Results for our project to investigate the Rubik's cube.



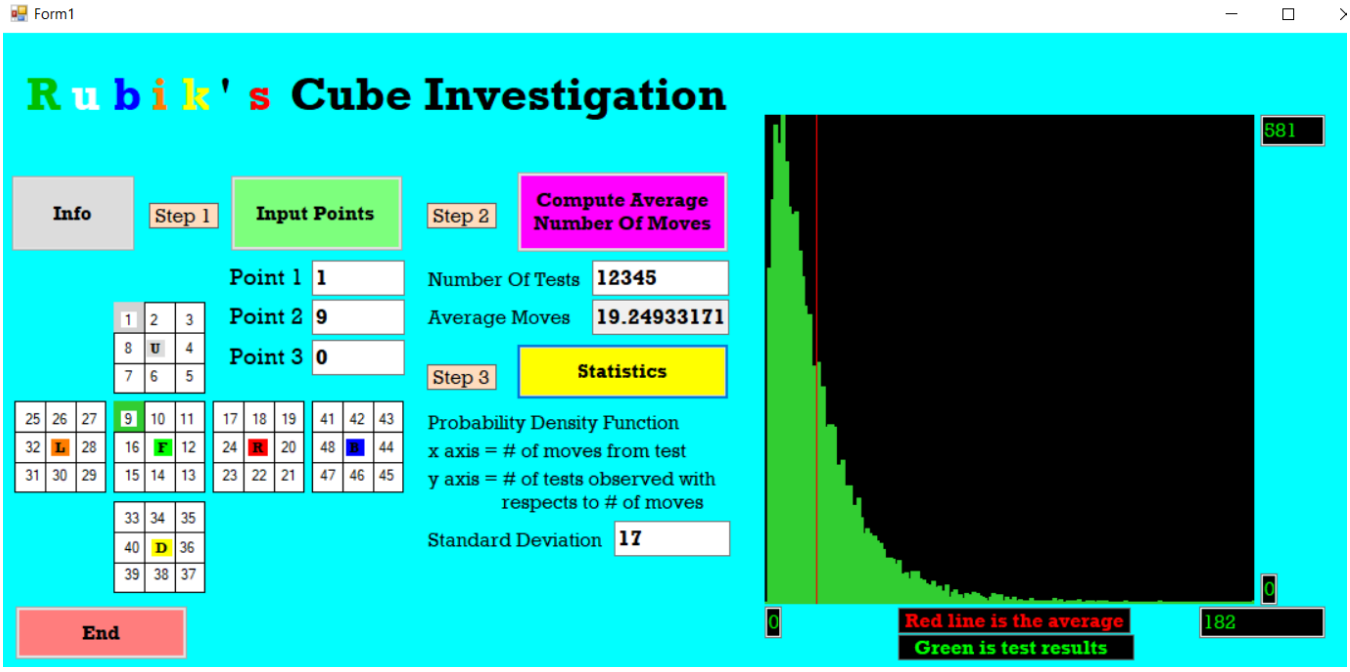
We can find the average number of random turns it takes for two selected points to appear on the same face. Points 1 and 9 with 12345 tests has an average of 19 moves.

Some examples:

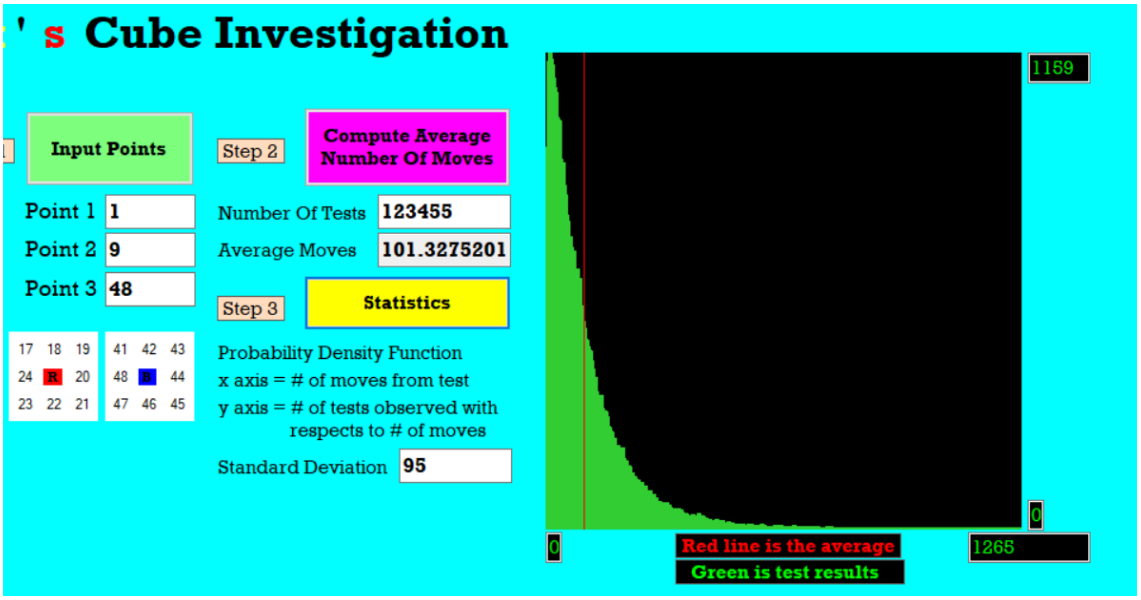
Number of Tests	Point 1 and 9	Point 1,9 and 48
100	16.2 - 21 on average	81 - 124 on average
1000	17.5 - 20.5 on average	94 - 107 on average
10000	18.7 - 19.3 on average	98 - 103 on average
100000	19.0 - 19.2 on average	101.2 - 101.4 on average
500000	19.07 - 19.09 on average	101.25 - 101.26 on average

The points will converge to an average number of turns for each and every test.

Project 1: The Statistics Part



Once the tests are run then Statistics is pressed to see the probability density function.
X-axis is the number moves it took for 2 or 3 points to get to the same side.
Y-axis is the number of test that displayed that particular number of moves.
X goes from 0 to the largest amount of moves found after running all tests.
Y goes from 0 to the most test found to have the same amount of moves.(ex 581 test had 6 moves)
We can see the display to not have an exactly bell curve. With bigger number of tests we will see that all different test points display more of an 1/x graph, seen below. It was very cool to see how much that different points had in common. I can also conclude that this data isn't very consistent as our standard deviation is very high compared to our average. Data is very skewed by some of the very few tests that had lots of moves recorded in it's single test. (Ex 1265 move=12.5(avg))



Project 1: Running tests to see results

1) Test 1 (2)3.8	2) Test 2 (1)3.8	(1)6.0 (2)3.8
(1)3.2 (2)7.4	**Up Face**	(1)8.2 (2)7.4
(1)5.5 (2)8.4	(1)8.5 (2)6.5	(1)10.0 (2)8.4

Running 500000 tests. Averages posted below in respect to test 1 or 2.

(1)X (2)22.0	(1)15.4 (2)24.4	(1)11.1 (2)13.3	(1)19.1 (2)15.3	(1)15.5 (2)24.4	(1)13.5 (2)15.3
(1)22.7 (2)23.3	**Left Face**	(1)13.7 (2)15.9	(1)19.1 (2)21.2	**Front Face**	(1)13.8 (2)20.6
(1)19.1 (2)18.8	(1)15.1 (2)20.6	(1)13.7 (2)13.8	(1)18.6 (2)16.1	(1)16.1 (2)22.6	(1)14.2 (2)15.7
			(1)17.1 (2)17.4	(1)17.5 (2)21.8	(1)15.8 (2)17.3
			(1)19.8 (2)23.8	**Down Face**	(1)16.9 (2)23.3
			(1)19.9 (2)19.6	(1)19.4 (2)26.6	(1)16.6 (2)19.4

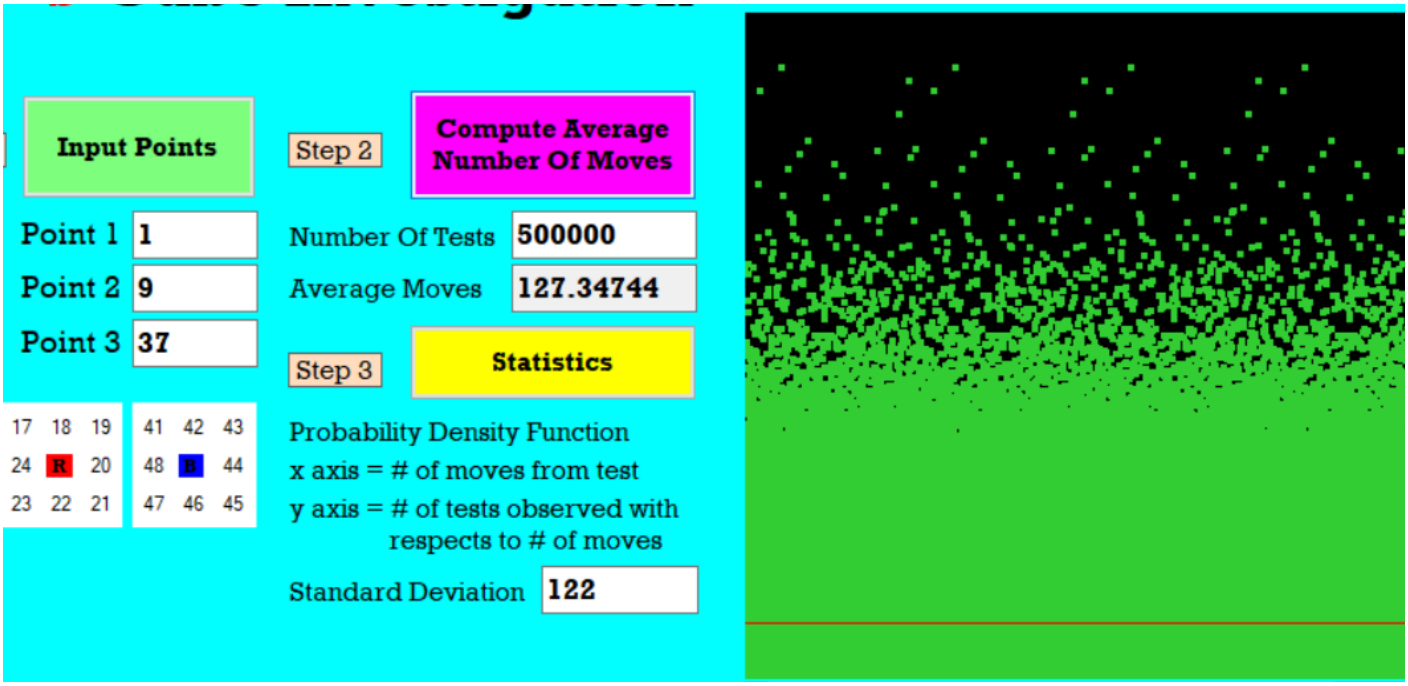
(1)14.6 (2)14.9	(1)17.0 (2)21.2	(1)19.0 (2)22.5	(1)11.3 (2)15.7	(1)15.6 (2)X	(1)X (2)15.6
(1)15.2 (2)18.4	**Right Face**	(1)18.6 (2)24.0	(1)12.9 (2)17.7	**Back Face**	(1)22.9 (2)19.1
(1)15.3 (2)15.0	(1)17.1 (2)22.6	(1)18.2 (2)18.8	(1)13.4 (2)15.6	(1)15.2 (2)24.3	(1)19.2 (2)15.8

This is the results after testing, Test 1(Upface top right) with every spot on the cube, highlighted in light blue. The second test shows the results after testing, Test 2(Upface top middle) with every spot on the cube. Ran 500,000 tests for all 91 tests done.

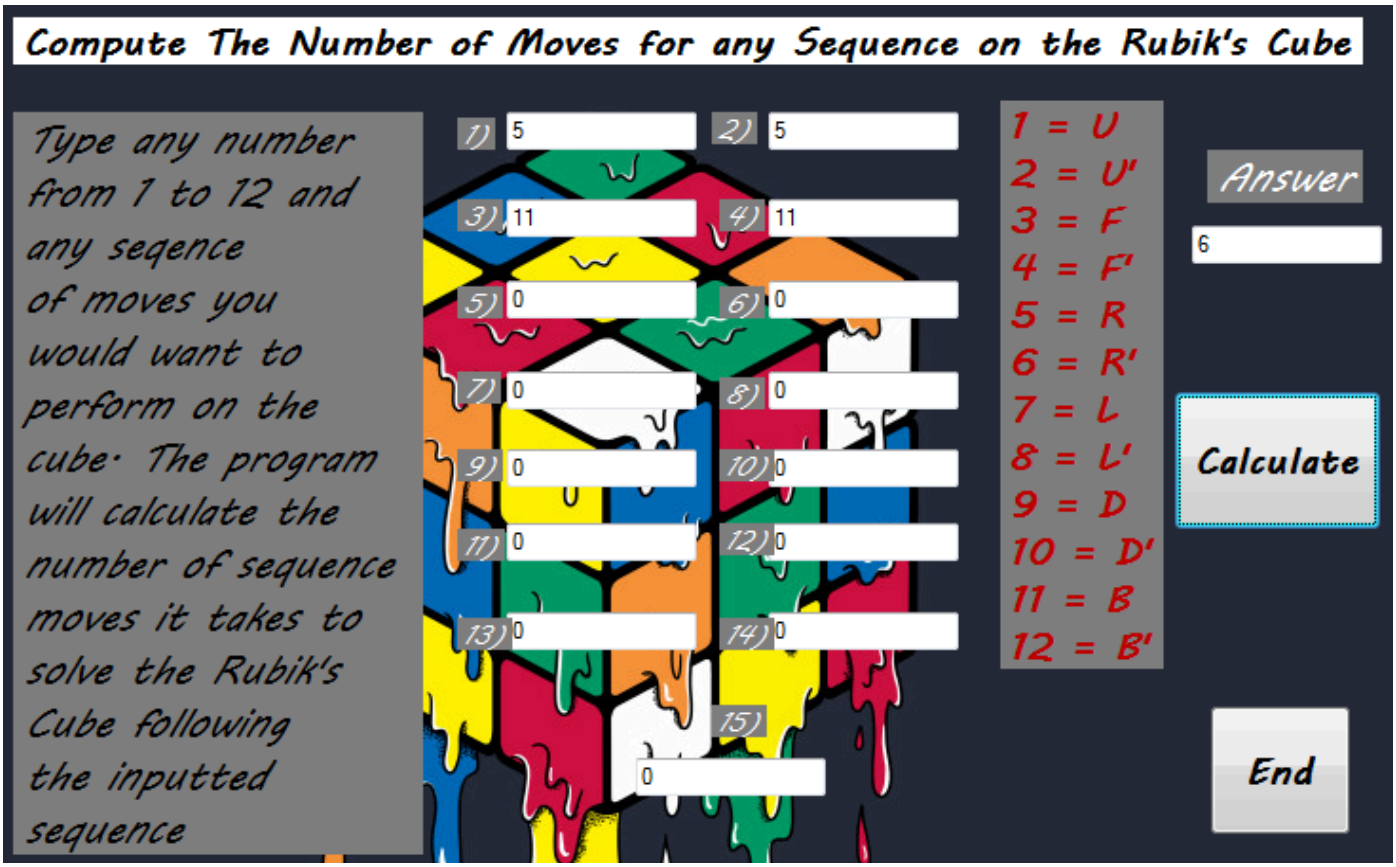
Note that when points start on the same face it isn't 0 because the program runs one move before checking, so if the first move separates the two points it continuous until they are back on the same face.

Before tests I thought I would see test with the same symmetry have the same results like testing 2 against the back right and left middle pieces. The right side had an average of 19.1 and the left side had an average of 17.7. This should be the same due to symmetry. So either my program isn't running all moves with even odds or VB doesn't like 500,000 tests.

Project 1



This is the result of 3 test points with 500,000 tests. I noticed that after approximately 150,000 tests the test points start to repeat. So doing more than 200,000 test won't be a proper estimate for average number of moves. It will get skewed by the first repetition of the first so many tests.



As seen in the above image, when the same 2 sequence of moves are repeated twice it takes exactly 6 repetitions of that sequence for the cube to come back to its solved state. The mathematical proof for variables is attached with this paper as well.

Experiments Of Group Theory

# of Moves	Move Sequence	Total time computed till solved state
1	U	4
2	U U	2
4	R R L L	2
4	R R B B	6
2		105
4	D L R F	24
6	R R L L U U	4
6	U U L R U D'	6
12	F F U L R' F F L' R U F F	3



The sequence is what has been entered into the program as elements of the Rubik's cube. All of the above sequences start from a solved state and when computed n amount of time it will become solved again. The n will be found in the output of the program. Above are some examples, where it says how many moves were made in one sequence, the particular move sequence in terms of cubing rotations(shown above) and the output, displaying the total time, n, it took to resolve the cube. We have examples of short and long sequences, which can vary in number of total computations until it resolves itself.