

# FINAL REPORT

## Of

# RUBIK'S CUBE SOLVER



School of Computer Science and Engineering  
Lovely Professional University.  
Phagwara, Punjab (India).

## INT 404 – ARTIFICIAL INTELLIGENCE

Submitted to:

GitHub link: -

Mrs. Shabnam

<https://github.com/SANIDHYADASH/INT404-AI-Rubik-s-Cube-Solver>

Submitted By:

Roll no	Registration No	Name of the Student	Marks	Signature
35	11802909	Utkarsh Patel		
36	11802843	Md. Saqulain		
39	11802151	Sanidhya Dash		
38	11802150	Debiprasad Sahoo		

## ABSTRACT

➔ Rubik's Cube is a widely popular mechanical puzzle that has attracted attention around the world because of its unique characteristics. As a classic brain-training toy well known to the public, Rubik's Cube was used for scientific research and technology development by many scholars. This paper provides a basic understanding of the Rubik's Cube and shows its mechanical art from the aspects of origin and development, characteristics, research status and especially its mechanical engineering design, as well as making a vision for the application in mechanism. First, the invention and origin of Rubik's Cube are presented, and then the special characteristics of the cube itself are analysed. After that, the present researches of Rubik's Cube are reviewed in various disciplines at home and abroad, including the researches of Rubik's Cube scientific metaphors, reduction algorithms, characteristic applications, and mechanism issues. Finally, the applications and prospects of Rubik's Cube in the field of mechanism are discussed.

## RELATED WORK

- Morwen B. Thistlethwaite is a mathematician who devised a clever algorithm for solving the Rubik's Cube in remarkably few moves. It is a rather complicated method, and therefore cannot be memorised. It is only practical for computers and not for humans. This algorithm is rather important from a theoretical standpoint however, as it has long been the method with the fewest number of moves.

Thistlethwaite's method differs from layer algorithms and corners first algorithms in that it does not place pieces in their correct positions one by one. Instead it works on all the pieces at the same time, restricting them to fewer and fewer possibilities until there is only one possible position left for each piece and the cube is solved.

The way it does this is by first doing a few moves until a position arises that can be solved without using quarter turns of the U and D faces (though half turns of U and D are still needed). It then proceeds to solve the cube without using U or D quarter turns by first moving to a position that does not need quarter turns of the F, B faces either. With these further restrictions a position is arrived at that does not need any quarter turns at all, and can hence be solved by half turns only. The cube then indeed gets solved using half turns only.

- Ryan Heise has developed a way of solving the cube based on Thistlethwaite's algorithm. He splits stages 2, 3 and 4 into two steps each (corners and edges separately) in order to make it possible for a human being to memorise.
- An algorithm for finding optimal solutions for Rubik's Cube was published in 1997 by Richard Korf. While, it had been known since 1995 that 20 was a lower bound on the number of moves for the solution in the worst case, it was proved in 2010 through extensive computer calculations that no configuration requires more than 20 moves. Thus 20 is a sharp upper bound on the length of optimal solutions. This number is known as **God's number**.

## IMPLEMENTATION

- ➔ A Rubik's Cube algorithm is an operation on the puzzle which reorients its pieces in a certain way. Mathematically the Rubik's Cube is a permutation group: an ordered list, with 54 fields with 6\*9 values (colours) on which we can apply operations (basic face rotations, cube turns and the combinations of these) which reorient the permutation group according to a pattern.
- ➔ The names of the facelet positions of the cube (letters stand for Up, Left, Front, Right, Back, and Down):

```
|*****|  
|*U1**U2**U3*|  
|*****|
```

```

|*U4**U5**U6*|
|*****|
|*U7**U8**U9*|
|*****|
*****|*****|*****|*****
*L1**L2**L3*|*F1**F2**F3*|*R1**R2**R3*|*B1**B2**B3*
*****|*****|*****|*****
*L4**L5**L6*|*F4**F5**F6*|*R4**R5**R6*|*B4**B5**B6*
*****|*****|*****|*****
*L7**L8**L9*|*F7**F8**F9*|*R7**R8**R9*|*B7**B8**B9*
*****|*****|*****|*****
|*****|
|*D1**D2**D3*|
|*****|
|*D4**D5**D6*|
|*****|
|*D7**D8**D9*|
|*****|

```

A cube definition string "UBL..." means that in position U1 we have the U-colour, in position U2 we have the B-colour, in position U3 we have the L colour etc. according to the order U1, U2, U3, U4, U5, U6, U7, U8, U9, R1, R2, R3, R4, R5, R6, R7, R8, R9, F1, F2, F3, F4, F5, F6, F7, F8, F9, D1, D2, D3, D4, D5, D6, D7, D8, D9, L1, L2, L3, L4, L5, L6, L7, L8, L9, B1, B2, B3, B4, B5, B6, B7, B8, B9.

So, for example, a definition of a solved cube would be  
UUUUUUUUURRRRRRRRRRFFFFFFFFFDDDDDDDDLLLLLLLLLLBBBBBBBBB

Solution string consists of space-separated parts, each of them represents a single move:

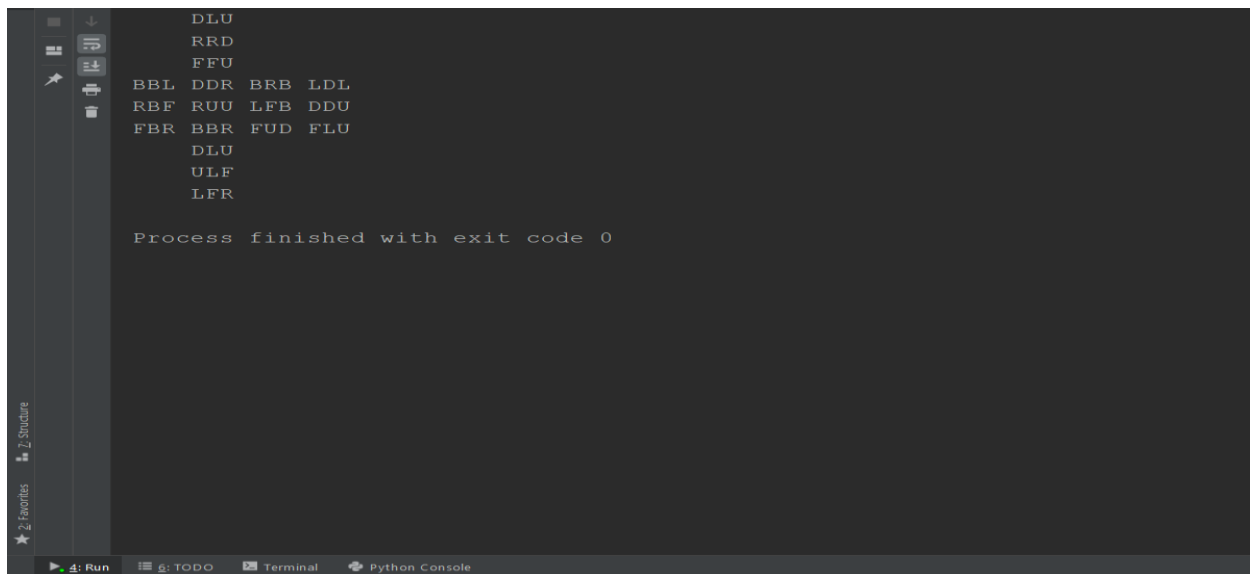
- A single letter by itself means to turn that face clockwise 90 degrees.
- A letter followed by an apostrophe means to turn that face counter clockwise 90 degrees.
- A letter with the number 2 after it means to turn that face 180 degrees.

e.g. **R U R' U R U2 R' U**

## RESULT / OUTPUT

→On executing the different files of the complete program for Rubik's cube solving, the result that will be obtained are:

**\*\*main.py:**



```
DLU
RRD
FFU
BBL DDR BRB LDL
RBF RUU LFB DDU
FBR BBR FUD FLU
DLU
ULF
LFR

Process finished with exit code 0
```

The screenshot shows a code editor with a dark theme. On the left, there is a sidebar with icons for file explorer, search, and other IDE features. The main area displays a list of Rubik's cube moves: DLU, RRD, FFU, BBL DDR BRB LDL, RBF RUU LFB DDU, FBR BBR FUD FLU, DLU, ULF, and LFR. Below the list, a message states 'Process finished with exit code 0'. The bottom status bar shows 'Run', 'TODO', 'Terminal', and 'Python Console' tabs.

**\*\*CubeSolve.py:**

```
↓
Solving:
  DLU
  RRD
  FFU
BBL DDR BRB LDL
RBF RUU LFB DDU
FBR BBR FUD FLU
  DLU
  ULF
  LFR
  DLU
  RRD
  FFU
BBL DDR BRB LDL
RBF RUU LFB DDU
FBR BBR FUD FLU
  DLU
  ULF
  LFR
cross
Cross:
  DFF
  DRR
  BRL
RLD LUU BFL DDB
```

```
↑
Cross:
  DFF
  DRR
  BRL
RLD LUU BFL DDB
FBB UUU FFB DDL
URF RUR URF LBF
  DLB
  DLB
  RLU
cross_corners
Corners:
  FRB
  RRL
  RRR
DFB UUU FBR DDL
FBB UUU FFR BDL
RDB UUU FBD BDF
  LLL
  LLD
  DFL
Second layer:
  DLB
  RRR
  RRR
FBB UUU FFD LDR
```

```

↑
↓
⏮
⏭
⏪
⏩
Second layer:
    DLB
    RRR
    RRR
FBB UUU FFD LDR
DBB UUU FFR DDF
DBB UUU FFB DBF
    LLL
    LLL
    LDR
Last layer edges
    LFB
    RRR
    RRR
FBB UUU FFR DDD
RBB UUU FFL DDD
RBB UUU FFB LDF
    LLL
    LLL
    DBD
Last layer corners -- position
    DDR
    RRR
    RRR
RBB UUU FFF DRB
LBB UUU FFD BDD

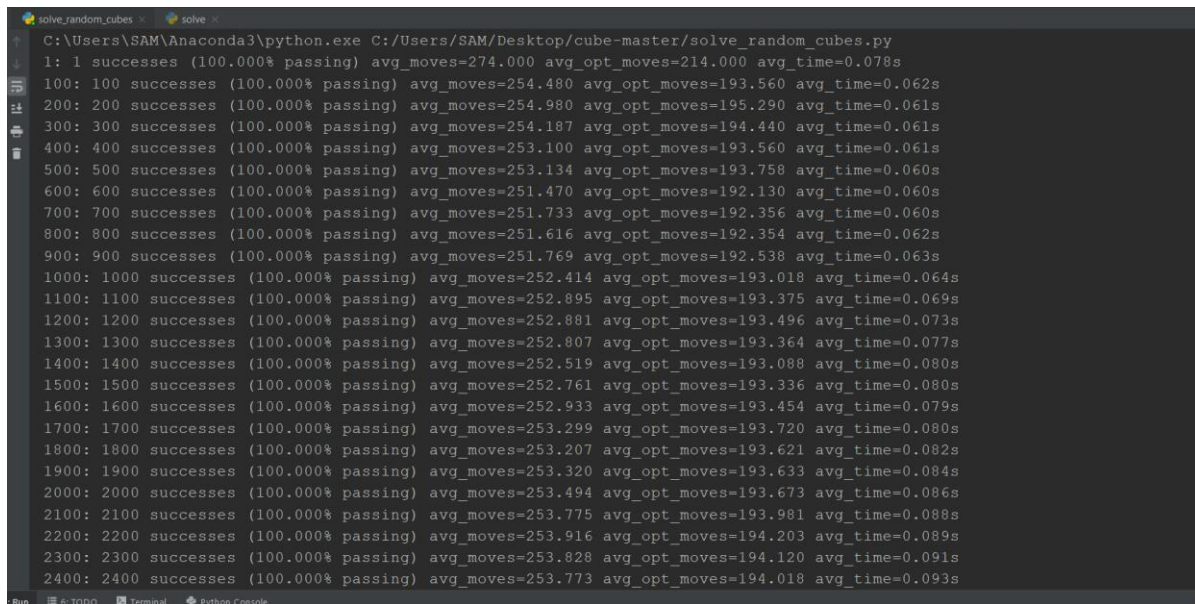
```

```

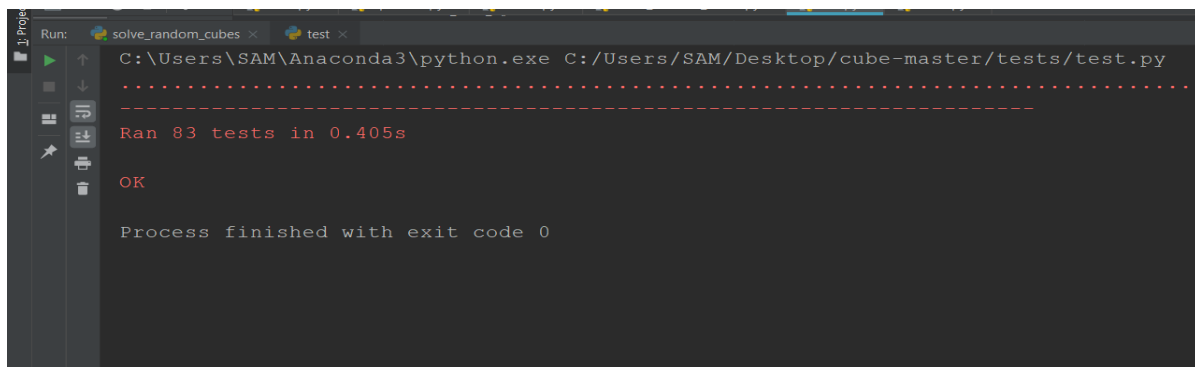
↑
↓
⏮
⏭
⏪
⏩
RRR
BBB UUU FFF DRD
DBB UUU FFF DDB
BBB UUU FFF DDD
    LLL
    LLL
    LLL
_handle_last_layer_state2
Solved
    FFF
    FFF
    FFF
LLL DDD RRR UUU
LLL DDD RRR UUU
LLL DDD RRR UUU
    BBB
    BBB
    BBB
222 moves: Di B B B D E L Ei Li D B Di R R Z B B B L L R R Zi Bi Ri B R Z B B Bi Ri B R Z B B B B D Bi Di Z B B B D Bi Di Z Z Z
B L Bi Li Bi Di B D Zi Zi B B B Bi Di B D B L Bi Li Z B B L Bi Li Bi Di B D Z Z B L Bi Li Bi Di B D Zi B B L Bi Li Bi Di B D Z
B B B Bi Di B D B L Bi Li Z X X F F D F R Fi Ri Di Xi Xi X X Z F Li Fi L D F Di Li F L F Zi Li Fi L D F Di Li F L F F F Li Fi
L D F Di Li F L F Xi Xi X X F F F Ri Fi R Fi Ri F F R F F F R Fi F R F F Ri F F Xi Xi X X Z Z Z Di Li Ri S Ri Ri S S Ri Fi
Fi R Si Si Ri Ri Si R Fi Fi L D
Process finished with exit code 0

```

**\*\*random\_cubes.py:**



```
C:\Users\SAM\Anaconda3\python.exe C:/Users/SAM/Desktop/cube-master/solve_random_cubes.py
1: 1 successes (100.000% passing) avg_moves=274.000 avg_opt_moves=214.000 avg_time=0.078s
100: 100 successes (100.000% passing) avg_moves=254.480 avg_opt_moves=193.560 avg_time=0.062s
200: 200 successes (100.000% passing) avg_moves=254.980 avg_opt_moves=195.290 avg_time=0.061s
300: 300 successes (100.000% passing) avg_moves=254.187 avg_opt_moves=194.440 avg_time=0.061s
400: 400 successes (100.000% passing) avg_moves=253.100 avg_opt_moves=193.560 avg_time=0.061s
500: 500 successes (100.000% passing) avg_moves=253.134 avg_opt_moves=193.758 avg_time=0.060s
600: 600 successes (100.000% passing) avg_moves=251.470 avg_opt_moves=192.130 avg_time=0.060s
700: 700 successes (100.000% passing) avg_moves=251.733 avg_opt_moves=192.356 avg_time=0.060s
800: 800 successes (100.000% passing) avg_moves=251.616 avg_opt_moves=192.354 avg_time=0.062s
900: 900 successes (100.000% passing) avg_moves=251.769 avg_opt_moves=192.538 avg_time=0.063s
1000: 1000 successes (100.000% passing) avg_moves=252.414 avg_opt_moves=193.018 avg_time=0.064s
1100: 1100 successes (100.000% passing) avg_moves=252.895 avg_opt_moves=193.375 avg_time=0.069s
1200: 1200 successes (100.000% passing) avg_moves=252.881 avg_opt_moves=193.496 avg_time=0.073s
1300: 1300 successes (100.000% passing) avg_moves=252.807 avg_opt_moves=193.364 avg_time=0.077s
1400: 1400 successes (100.000% passing) avg_moves=252.519 avg_opt_moves=193.088 avg_time=0.080s
1500: 1500 successes (100.000% passing) avg_moves=252.761 avg_opt_moves=193.336 avg_time=0.080s
1600: 1600 successes (100.000% passing) avg_moves=252.933 avg_opt_moves=193.454 avg_time=0.079s
1700: 1700 successes (100.000% passing) avg_moves=253.299 avg_opt_moves=193.720 avg_time=0.080s
1800: 1800 successes (100.000% passing) avg_moves=253.207 avg_opt_moves=193.621 avg_time=0.082s
1900: 1900 successes (100.000% passing) avg_moves=253.320 avg_opt_moves=193.633 avg_time=0.084s
2000: 2000 successes (100.000% passing) avg_moves=253.494 avg_opt_moves=193.673 avg_time=0.086s
2100: 2100 successes (100.000% passing) avg_moves=253.775 avg_opt_moves=193.981 avg_time=0.088s
2200: 2200 successes (100.000% passing) avg_moves=253.916 avg_opt_moves=194.203 avg_time=0.089s
2300: 2300 successes (100.000% passing) avg_moves=253.828 avg_opt_moves=194.120 avg_time=0.091s
2400: 2400 successes (100.000% passing) avg_moves=253.773 avg_opt_moves=194.018 avg_time=0.093s
```



```
C:\Users\SAM\Anaconda3\python.exe C:/Users/SAM/Desktop/cube-master/tests/test.py
.....
-----
Ran 83 tests in 0.405s

OK

Process finished with exit code 0
```

## IMPORTANT LIBRARIES USED

NumPy → Used for using rotation functions

Random → Used for scrambling the cube for random colour positions

Sys → System-specific parameters and functions. This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter.

Time → It provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code.



Textwrap → It can be used for wrapping and formatting of plain text and it provides formatting of text by adjusting the line breaks in the input paragraph.

## **TEAM RESPONSIBILITIES**

Utkarsh -> Make the representation of the cube for solving

Md. Saqulain -> Use the algorithm for solving the Rubik's cube

Sanidhya Dash -> Use the algorithm for solving the Rubik's cube

Debiprasad Sahoo -> Use the algorithm for solving the Rubik's cube

## **REFERENCES**

<https://ruwix.com/the-rubiks-cube/algorithm/>

<https://rubiks-cube-solver.com/>

<https://fulmicoton.com/posts/rubix/>

<https://towardsdatascience.com/learning-to-solve-a-rubiks-cube-from-scratch-using-reinforcement-learning-381c3bac5476>

[https://en.wikipedia.org/wiki/Rubik%27s\\_Cube#Conception\\_and\\_development](https://en.wikipedia.org/wiki/Rubik%27s_Cube#Conception_and_development)

<https://www.jaapsch.net/puzzles/thistle.htm>