This is how I planned for this app.

Excuse me for the messy handwriting (when I think and write rough) 🙏

You probably don't want to read anything from it, but just try to go up to the last page, you will get some things for sure, thank you.

Rubrik's Cube Solver

Using OOP:

Objective: 1) Build a cube in OO model.

2) Logic to manually rotate cube faces.

3) Scramble logic.

4) Solve logic.

Understanding cube:

Six (6) faces, Nine (9) stickers each face, 54 stickers in total.

To know the color of stickers on each face, use a string. Ex: bbbbbbbbb (All blue colored face).

999999999 (green).

Colors and faces (General format):

1) ~~Green~~ White - Up

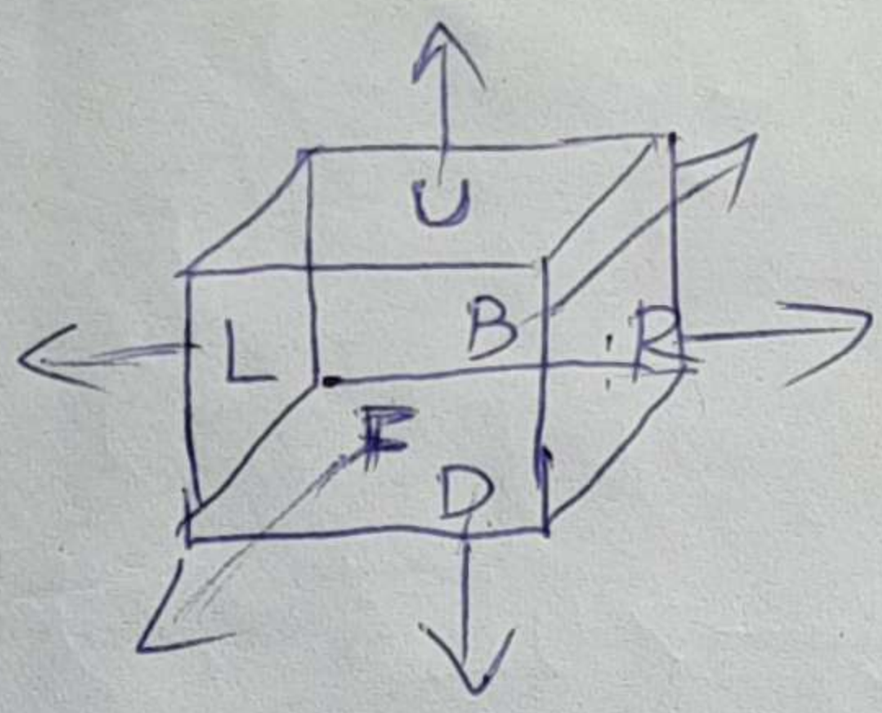
~~Blue~~ 2) Down - Yellow

3) Front - Green

4) Back - Blue.

5) Left : Orange

6) Right : Red.



W -> U

Y -> D

G -> F

B -> B

O -> L

R -> R

~~Hea~~ Use class ~~for~~ cube construction:

this.faces = {U: , D: , F: , ... } (arrays)

Rotation Logic:

In a cube, we rotate one of the faces, in either clockwise or counter clockwise

0 1 2
3 4 5
6 7 8

clock wise →

6 3 0
7 4 1
8 5 2

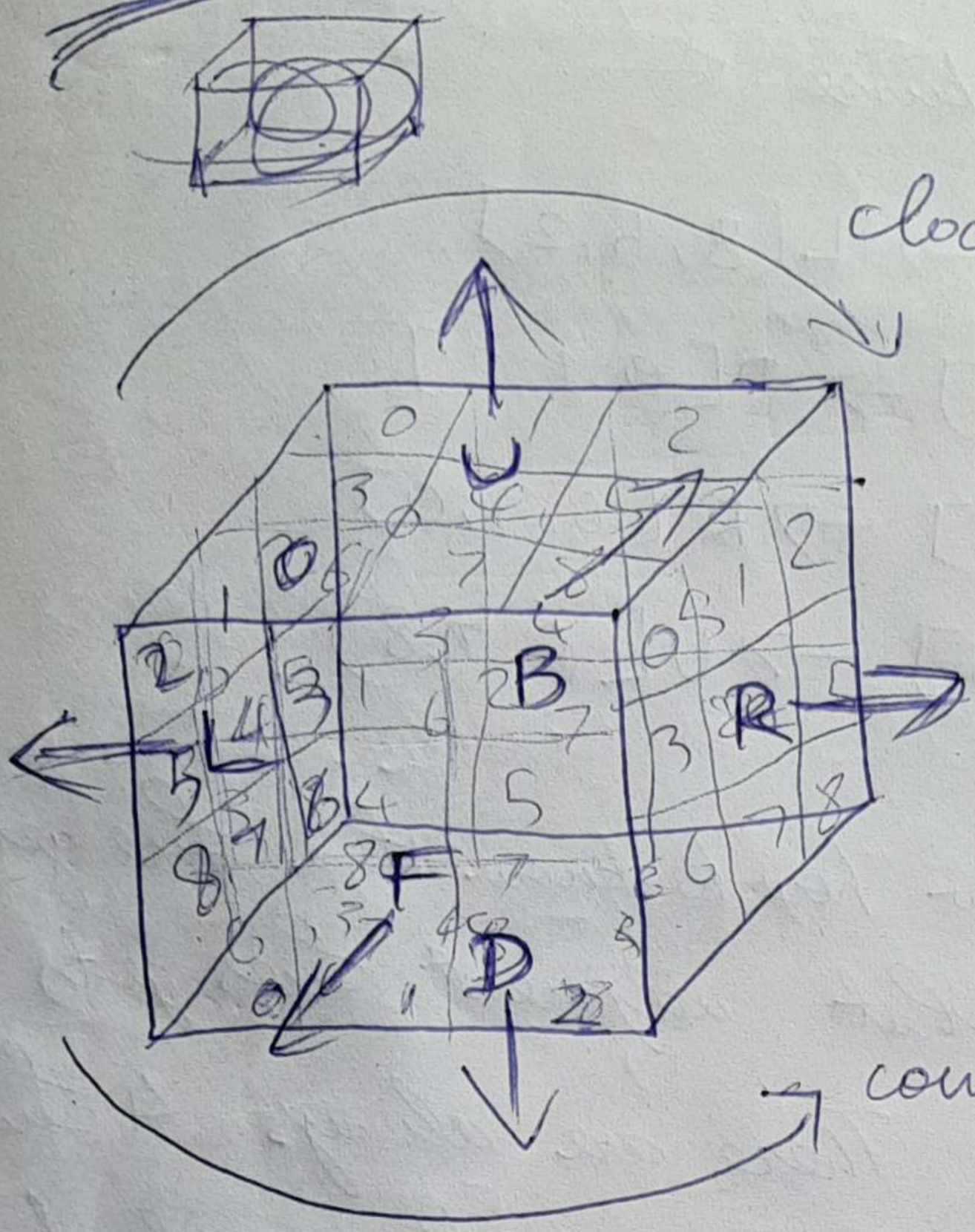
Now a rotation also affects neighbors / adjacent faces. If front is rotated, U, R, D, L gets changed. w counter clockwise.

U[6,7,8] turns into L[2,5,8],

L[2,5,8] → D[.,.,.], D[ ] → R[ ]

R[.,.,.] → U[.,.,.].

RuBik's CUBE SOLVER



Clockwise F →

U[6,7,8] → R[0,3,6],

R[0,3,6] → D[2,1,0]

D[2,1,0] → L[8,5,2].

L[~~2,5,8~~] → U[6,7,8].
8,5,2

~~counter clockwise F ←~~

~~U[6,7,8] ← → L[2,5,8]~~

~~R[0,3,6] →~~

~~D[2,1,0] →~~

~~[8,5,2] →~~

See next pg. →

Counter clockwise ←

U[6,7,8] → L[8,5,2]

L[2,5,8] → D[0,1,2]

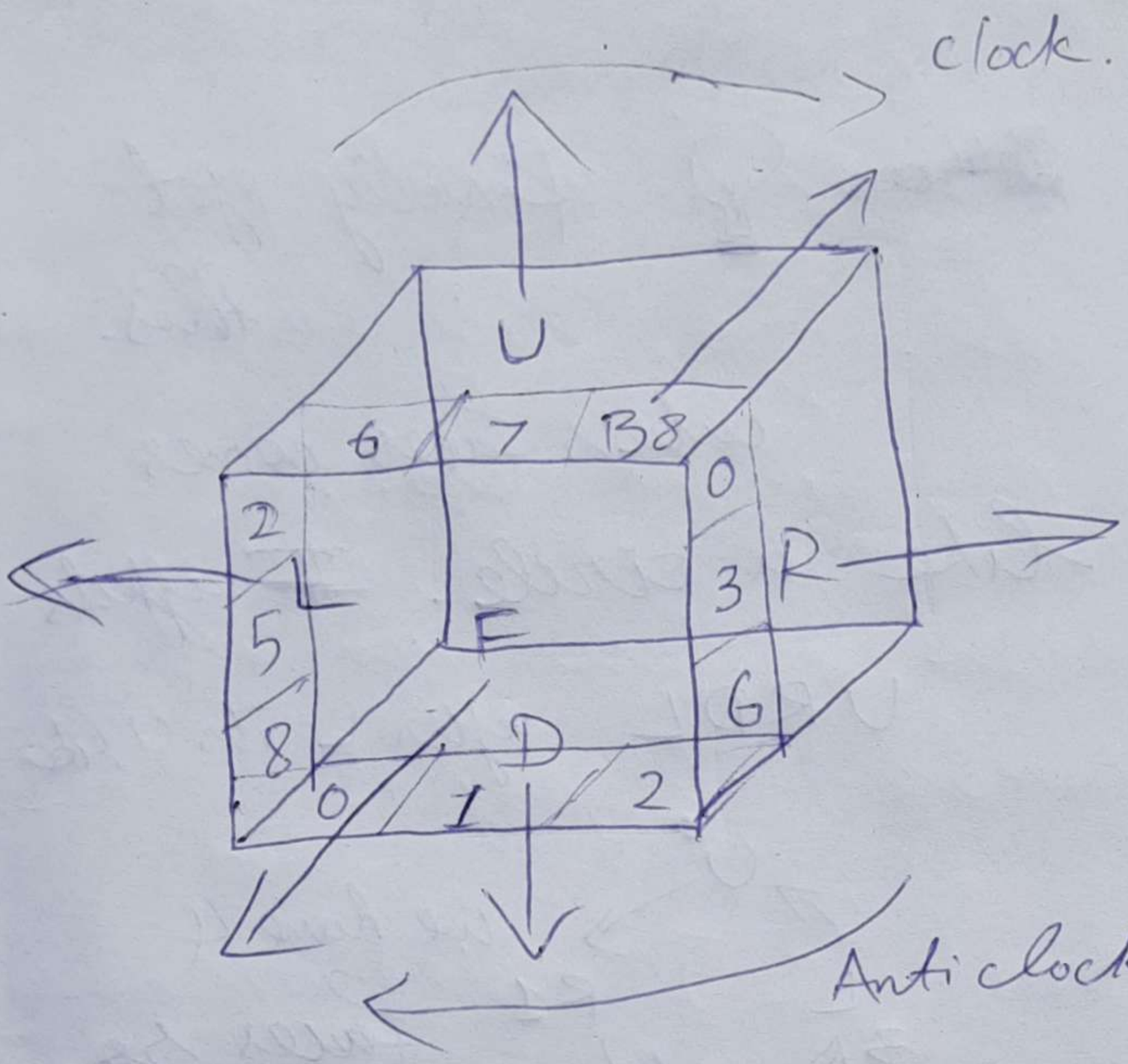
D[0,1,2] → R[6,3,0]

R[0,3,6] → U[6,7,8].

~~Now for~~ Now how to know if a face gets turned. What adjacent faces gets affected? There are always 4 faces affected. Create a facemap object to map it.

For ex:

```
facemap={ F:[{face:'U', indices:[6,7,8]},
            {face:'R', indices:[0,3,6]},
            :
            ],
B:
:
U:
D:
L:
R:
};
```



Now how to figure rotation? Face order.

0 :
1 :
2 :
3 :
4 :
5 :

~~Indices of~~
~~faces in this face~~

So if F gets rotated

~~U turns into~~
R gets updated by U
D gets updated by R
⋮

So how to change U's values into R.

~~from~~ this.~~face~~s[~~from~~to face] = this.faces.[from face]

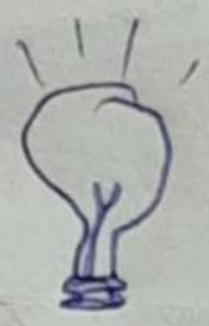
this.faces.R becomes this this.faces.U.

5

See next pg →

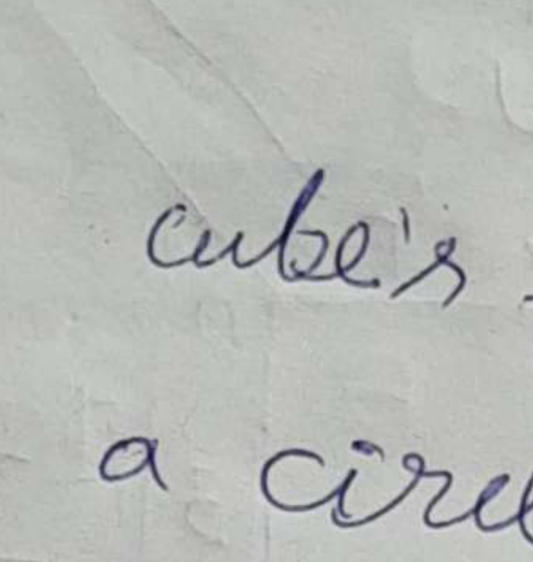
Why is this hard? :(

U
R
D
L
F
B



finally got this.

cube's faces comes like a circle. ~~F~~ affects URDL after L its U, like



we have 4 faces so. in clockwise.

U ~~get~~ gets updated by L,

So 0=3, how to bring into circle? use %4.

1 = 0,
2 = 1,
3 = 2.

~~If~~ So, (i+3)%4;

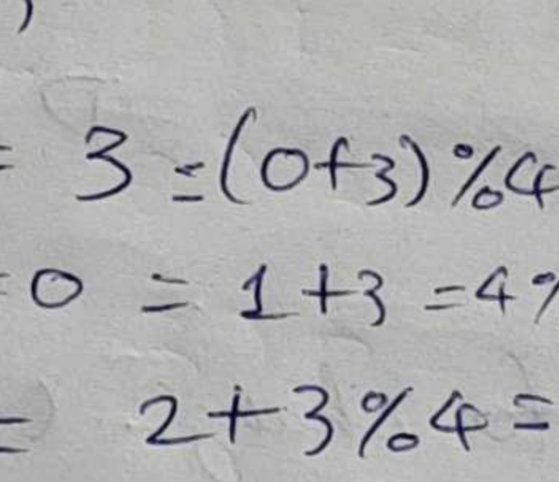
~~6~~ 0 = 3 = (0+3)%4

1 = 0 = 1+3 = 4%4 = 0.

2 = 2+3%4 = 1

3 = 3+3%4 = 2.

for counter clockwise,



0 = 1,
1 = 2,
2 = 3,
3 = 0.

how to get this?, => (i+1)%4.

Rotation logic is done.

Scramble ( ):

- Do a few random moves (let's say 15).

Use math.random(). Get a value from 0 to 5 for face and 0 to 1 for direction. For example: If you get (~~0~~ 3 and 0) - 3 is

U 0
R 1
D 2
L 3 →
F 4
B 5

0 - clockwise
1 - anticlockwise

~~this faces. L 0 and~~

rotateFace(L, clockwise).

- A rotation of face L in clockwise is done.
- Use for loop to do 15 random moves.

Solve():

I'm no genius in solving rubik's cube. So, let's do what CS programmers do..

Just reverse whatever you did on scramble (easy).

- Use stack (array) = []. Store all moves one by one in scramble().

Iterate from reverse of the array and reverse the direction and you've solved the cube.

Example: Scramble: 1) F → clockwise,
2) B → anti clockwise,
3) D → anti clockwise.

Solve: 1) D → clockwise
2) B → clockwise
3) F → anticlockwise

BINGO!

Get SvgCube():

We have to show the picture (just yapping won't be enough).

A cube has 9 stickers each side.

6 sides 6x9 = 54 stickers of 6 colors.

Let's have a 54 character string =

"wwwwwwwwwyyyyyyyyyrrr...ooo...bbb...ggg..."

white yellow red Blue green..

Now slice the string into six substring.

wwww... rrrr... gggg... yyyy... oooo... bbbb...

white → Up face

red → Right face

green → Front

yellow → Down

orange → Left

blue → Back

For every rotateFace(face, direction). This colors get updated.

That's it