# Rubix Python Solver

## Nodes/States

The state of the cube

* 2x2 Cube
  + Each row is a list (2 values per row)
  + Each side is a list of row lists (2 values/rows per side)
  + Complete cube is represented as a 6x2x2 (6 faces, 2 rows, 2 values)
  + [[Front], [Back], [Left], [Right], [Top], [Bottom]]... where [Front] = [[1,2], [3,4]]

Ex: Front

[[1,2], [3,4]]

| 1 | 2 |
| --- | --- |
| 3 | 4 |

## Paths/Operations

For 2x2 cube: 6 x 2 = 12 operations

**\*FROM POV OF FRONT FACE**

* Top row
  + Turn left + turn right
* Bottom row
  + Turn left + turn right
* Left column
  + Turn down + turn up
* Right column
  + Turn down + turn up
* Front face
  + Turn clockwise + turn counterclockwise
* Back face
  + Turn clockwise + turn counterclockwise

For 3x3 cube: 6 x 2 = 12 operations

**\*FROM POV OF FRONT FACE… These are all non-redundant moves excluding middle row/column moves**

* Top row
  + Turn left + turn right
* Bottom row
  + Turn left + turn right
* Left column
  + Turn down + turn up
* Right column
  + Turn down + turn up
* Front face
  + Turn clockwise + turn counterclockwise
* Back face
  + Turn clockwise + turn counterclockwise
* ~~Middle row~~
  + ~~Turn left + turn right~~
* ~~Middle column~~
  + ~~Turn down + turn up~~

How does this translate into matrix representation of the cube?

* For all operations, switch lists in matrix + rotate edge face

### 

### Example 1

* [[[y,y],[y,y]], [[w,w],[w,w]], [[g,g],[g,g]], [[b,b],[b,b]], [[r,r],[r,r]], [[o,o],[o,o]]]

|  | | **TP** | 1 | 2 |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 |  |
| **LT** |  |  | **FT** |  |  | **RT** |  |
| 2 | 4 |  | 1 | 2 |  | 3 | 1 |
| 1 | 3 |  | 3 | 4 |  | 4 | 2 |
|  |  |  |  |  |  |  |  |
|  | | **BM** | 4 | 3 |  |  | |
|  | 2 | 1 |  |

**BK**

| 2 | 1 |
| --- | --- |
| 4 | 3 |

* **Operation: top row left turn**
* [[[b,b],[y,y]], [[w,g],[g,w]], [[g,y],[g,y]], [[w,b],[w,b]], [[r,r],[r,r]], [[o,o],[o,o]]]

|  | | **TP** | 1 | 2 |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 |  |
| **LT** |  |  | **FT** |  |  | **RT** |  |
| 2 | 4 |  | 1 | 2 |  | 3 | 1 |
| 1 | 3 |  | 3 | 4 |  | 4 | 2 |
|  |  |  |  |  |  |  |  |
|  | | **BM** | 4 | 3 |  |  | |
|  | 2 | 1 |  |

**BK**

| 2 | 1 |
| --- | --- |
| 4 | 3 |

### Example 2

* [[[o,y],[o,y]], [[w,r],[w,r]], [[g,g],[g,g]], [[b,b],[b,b]], [[y,r],[y,r]], [[o,w],[o,w]]]

|  | | **TP** | 1 | 2 |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 |  |
| **LT** |  |  | **FT** |  |  | **RT** |  |
| 2 | 4 |  | 1 | 2 |  | 3 | 1 |
| 1 | 3 |  | 3 | 4 |  | 4 | 2 |
|  |  |  |  |  |  |  |  |
|  | | **BM** | 4 | 3 |  |  | |
|  | 2 | 1 |  |

**BK**

| 2 | 1 |
| --- | --- |
| 4 | 3 |

* **Operation: top row left turn**
* [[[b,b],[o,y]], [[g,g],[w,r]], [[g,o],[g,y]], [[w,b],[r,b]], [[y,y],[r,r]], [[o,w],[o,w]]]

|  | | **TP** | 1 | 2 |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 |  |
| **LT** |  |  | **FT** |  |  | **RT** |  |
| 2 | 4 |  | 1 | 2 |  | 3 | 1 |
| 1 | 3 |  | 3 | 4 |  | 4 | 2 |
|  |  |  |  |  |  |  |  |
|  | | **BM** | 4 | 3 |  |  | |
|  | 2 | 1 |  |

**BK**

| 2 | 1 |
| --- | --- |
| 4 | 3 |

### Example 3

* [[[y,y],[y,y]], [[w,w],[w,w]], [[g,g],[g,g]], [[b,b],[b,b]], [[r,r],[r,r]], [[o,o],[o,o]]]

|  | | **TP** | 1 | 2 |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 |  |
| **LT** |  |  | **FT** |  |  | **RT** |  |
| 2 | 4 |  | 1 | 2 |  | 3 | 1 |
| 1 | 3 |  | 3 | 4 |  | 4 | 2 |
|  |  |  |  |  |  |  |  |
|  | | **BM** | 4 | 3 |  |  | |
|  | 2 | 1 |  |

**BK**

| 2 | 1 |
| --- | --- |
| 4 | 3 |

* **Operation: front face clockwise turn**
* [[[y,y],[y,y]], [[w,w],[w,w]], [[g,g],[o,o]], [[b,b],[r,r]], [[r,r],[g,g]], [[o,o],[b,b]]]

|  | | **TP** | 1 | 2 |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 |  |
| **LT** |  |  | **FT** |  |  | **RT** |  |
| 2 | 4 |  | 1 | 2 |  | 3 | 1 |
| 1 | 3 |  | 3 | 4 |  | 4 | 2 |
|  |  |  |  |  |  |  |  |
|  | | **BM** | 4 | 3 |  |  | |
|  | 2 | 1 |  |

**BK**

| 2 | 1 |
| --- | --- |
| 4 | 3 |

## Example 4

* [[[o,y],[o,y]], [[w,r],[w,r]], [[g,g],[g,g]], [[b,b],[b,b]], [[y,r],[y,r]], [[o,w],[o,w]]]

|  | | **TP** | 1 | 2 |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 |  |
| **LT** |  |  | **FT** |  |  | **RT** |  |
| 2 | 4 |  | 1 | 2 |  | 3 | 1 |
| 1 | 3 |  | 3 | 4 |  | 4 | 2 |
|  |  |  |  |  |  |  |  |
|  | | **BM** | 4 | 3 |  |  | |
|  | 2 | 1 |  |

**BK**

| 2 | 1 |
| --- | --- |
| 4 | 3 |

* **Operation: front face clockwise turn**
* [[[o,y],[o,y]], [[w,r],[w,r]], [[g,g],[g,g]], [[b,b],[b,b]], [[y,r],[y,r]], [[o,w],[o,w]]]

|  | | **TP** | 1 | 2 |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 |  |
| **LT** |  |  | **FT** |  |  | **RT** |  |
| 2 | 4 |  | 1 | 2 |  | 3 | 1 |
| 1 | 3 |  | 3 | 4 |  | 4 | 2 |
|  |  |  |  |  |  |  |  |
|  | | **BM** | 4 | 3 |  |  | |
|  | 2 | 1 |  |

**BK**

| 2 | 1 |
| --- | --- |
| 4 | 3 |

## Operations to Code

* Track keep graph/record of state matrices
  + Check if proposed operation yields state that has already been reached (aka state that is in the graph or list of previous state matrices)
* Use temp variables

Cube = [[[y,y],[y,y]],[[w,w],[w,w]],[[g,g],[g,g]],[[b,b],[b,b]],[[r,r],[r,r]],[[o,o],[o,o]]]

Front\_1\_temp = y = cube[0][0][0]

Front\_2\_temp = y = cube[0][0][1]

Front\_3\_temp = y = cube[0][1][0]

Front\_4\_temp = y = cube[0][1][1]

Back\_1\_temp = w = cube[1][0][0]

Back\_2\_temp = w = cube[1][0][1]

Back\_3\_temp = w = cube[1][1][0]

Back\_4\_temp = w = cube[1][1][1]

Left\_1\_temp = g = cube[2][0][0]

Left\_2\_temp = g = cube[2][0][1]

Left\_3\_temp = g = cube[2][1][0]

Left\_4\_temp = g = cube[2][1][1]

Right\_1\_temp = b = cube[3][0][0]

Right\_2\_temp = b = cube[3][0][1]

Right\_3\_temp = b = cube[3][1][0]

Right\_4\_temp = b = cube[3][1][1]

Top\_1\_temp = r = cube[4][0][0]

Top\_2\_temp = r = cube[4][0][1]

Top\_3\_temp = r = cube[4][1][0]

Top\_4\_temp = r = cube[4][1][1]

Bottom\_1\_temp = o = cube[5][0][0]

Bottom\_2\_temp = o = cube[5][0][1]

Bottom\_3\_temp = o = cube[5][1][0]

Bottom\_4\_temp = o = cube[5][1][1]

* There needs to be a function for each operation (12)
  + In each function the temp variables are assigned the same value, but then the matrix values are assigned to different temp variables according to which operation took place
  + Ex: **top row right turn**

First… Front\_1\_temp = cube[0][0][0]

Then… cube[0][0][0] = Left\_1\_temp

* Now we code out each operation!
* Then we create functions/loops for each search method
  + Start with DFS and BFS
  + Something like,

dfs\_func(input\_state):

…

return (number\_of\_moves, end\_state)

## 

## Process Flow

\*Create randomize functionality to create random yet plausible input states/matrices

* Do this with random number generators and for loops/while loops
* Ex: pick random number 1-100 for number of operations, then pick random numbers 1-12 for each operation

### DFS

With a starting cube/matrix, go down operations branch recursively until either a repeat state is reached or the goal state is reached

* Input state matrix
* Go down branch until repeat state/matrix is reached
  + 12 operations in standard order (ordered list)
  + Ex: 1) top row left turn -> 2) top row left turn -> 3) top row left turn -> 4) top row left turn… repeat state/matrix is reached

### BFS

## Misc. Logic

* DOES IT MAKE SENSE TO KEEP A LIST OF ALL PRIOR STATE MATRICES?
  + Loop through this after each attempted operation to check if it is a repeat
  + Downside is that it might be too large and slow the program down
  + WHAT WOULD BE AN ALTERNATIVE?
* Visualize numerical assignments for a 3x3 cube
  + Backside not shown, FT = front, LT = left, RT = right, TP = top, BM = bottom

|  | | | **TP** | 1 | 2 | 3 |  |  | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 4 | 5 | 6 |  |
|  | 7 | 8 | 9 |  |
| **LT** |  |  |  | **FT** |  |  |  | **RT** |  |  |
| 3 | 6 | 9 |  | 1 | 2 | 3 |  | 7 | 4 | 1 |
| 2 | 5 | 8 |  | 4 | 5 | 6 |  | 8 | 5 | 2 |
| 1 | 4 | 7 |  | 7 | 8 | 9 |  | 9 | 6 | 3 |
|  |  |  |  |  |  |  |  |  |  |  |
|  | | | **BM** | 9 | 8 | 7 |  |  | | |
|  | 6 | 5 | 4 |  |
|  | 3 | 2 | 1 |  |

* Visualize numerical assignments for a 2x2 cube
  + Backside not shown, FT = front, LT = left, RT = right, TP = top, BM = bottom

|  | | **TP** | 1 | 2 |  |  | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 |  |
| **LT** |  |  | **FT** |  |  | **RT** |  |
| 2 | 4 |  | 1 | 2 |  | 3 | 1 |
| 1 | 3 |  | 3 | 4 |  | 4 | 2 |
|  |  |  |  |  |  |  |  |
|  | | **BM** | 4 | 3 |  |  | |
|  | 2 | 1 |  |

* Corner dependencies
  + Ex (3x3): If front top left is yellow, then left top right must be yellow and top bottom left must be yellow
  + Ex (3x3): If front middle right is green, then right middle left must be green