Read the comments in the file for more detailsthis is the just the rubriks and the basics of the same

Basics about the structure:

>the structure that we are using is a 3D tuple list

>it includes six faces and each face have the desired no of cubies(small colored elements)

>the cube is initialised using a string that is of length n^n\*6 (n - the dimention of the cube )

> the cube can be initialised using the user given string as well as a new cube can also be constructed

> the code as of now is optimised for a 3\*3 cube

basics about the solved checker:

> the checker moves from one face to another

> it basically checks every face each time is is initiated

> it optimises the set rules to check for any discrepancy in the face of the cube

> if any face is found to be not solved at that instance it breaks ans false value is returned

basics about stringify:

> it basically cnverts the cube structure into the pre mentioned string data

basics about shuffler:

> it uses the random function to choose which ror/coluumn to be shuffled

> there are a possiblilty of 18 moves in total when each row and column transformation is taken in consideration

> the action (the pre defined transfromations) is chosen at random and cannot be interfered in between

basics about the dube displayer:

> it utilises a basic construct to create the gap between the zeroth and sixth face to give a more understandable display of the cube

> it goes face by face using the construct of the tuple list

> the colors of the cube are displayed using abbreviated colors (ex - 'r' for red )

basics about the twists:

> the twists are done on the pre assumption that the viewer is viewing the cube from the second face (in the structure that we made)

basics about horizontal twist:

> first it checks if the row no is in bounds

> then it checks which direction to twist( based on '0' and '1')

> then it interchanges the data between the affected cube faces

> if the row that is needed to be changed is the '0' or 'n-1' row then the face connected to the edge also needs to be transposed

> basic transpose of the facs is done according to need (right or left transpose to be exact)

Basics about the vertical twists:

> the basics is same as horizontal

> main diffrence is that when we do vertical changes we have to interchange the data between every row affected and as a result the time complexity increases.

Basics about the side twists :

> same basics as the above two

Basics about the Heuristic Builder:

> what is does is it takes the un shuffled cube and does each action on it and stores the obtained cube as a string with how many steps (actions) it took to reach that state

> here we use a library tqdm which helps us keep a track of the build process (it shows a progress bar along with the process count)

> here the heuristic is stored in a python dictionary that utilizes the concepts of a set i.e. no duplicates are allowed in the database

> the builder checks for dupliactes and if the state can be reached using less no of steps it upadtes the value of the steps needed to reach the state in the dictionary itself.

> it continues until the no of max moves is obatined

> the node count is calculates as sum of [len(actions) \*\* (x + 1) for x in range(max\_moves + 1)] {here actions -> 18 , max\_moves -> pre defined }

> the builder then one by one does operations on the current state of the cube and stores the data obtained in the dictionary

> it checks if the states obtained by the operations are in the DB or not

-> if not present it adds them to the DB

-> if present it checks if the moves required is less than or equal to the DB

- if true it updates

- else it leaves the DB as it is

> it also utilizes a que to keep track of the next state to perfprm operations on

-> first state is the unshuffled state

-> the new states obtained are then added to the que

-> it helps in stopping the left recursion of the tree (just for refrence not actuallu happening)

->