GREEDY

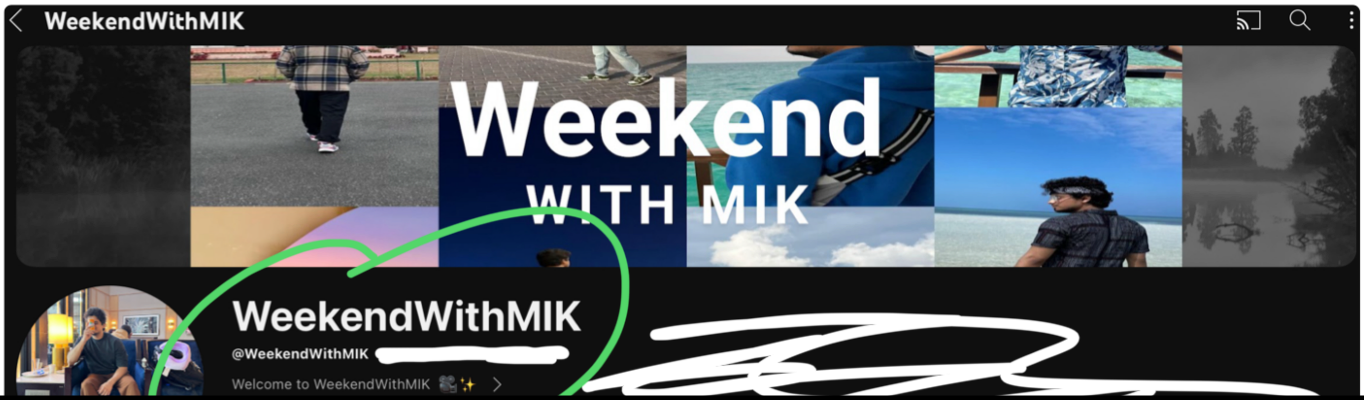
VIDEO - 40

Leetcode
-1007
Medium

Facebook
Instagr m → codestorywithMIK

Twitt → cswithMIK

→ codestorywithMIK



WeekendWithMIK

@WeekendWithMIK [redacted]

Welcome to WeekendWithMIK 🎥✨ >

[redacted]

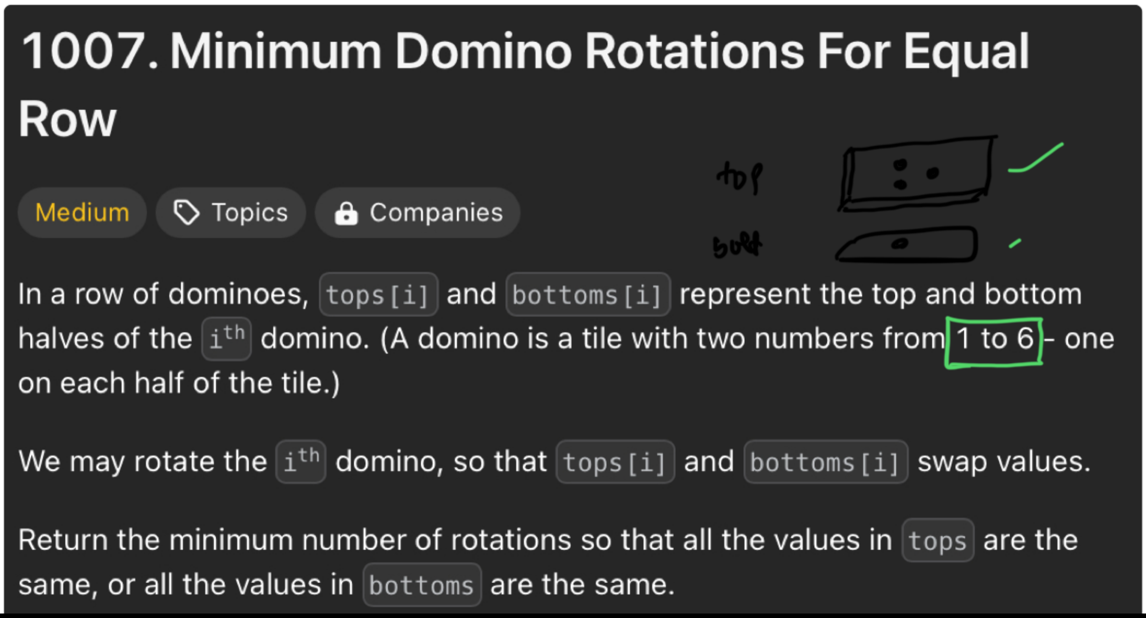
Try this channel to
see Life Behind The Scenes ...

Motivation Of the Day :-

Look around yourself, you have a home and a room where you can study. Think about them who are not blessed with such facilities. So, make the most of what you've got.

Make it count. You got this.

MIK...



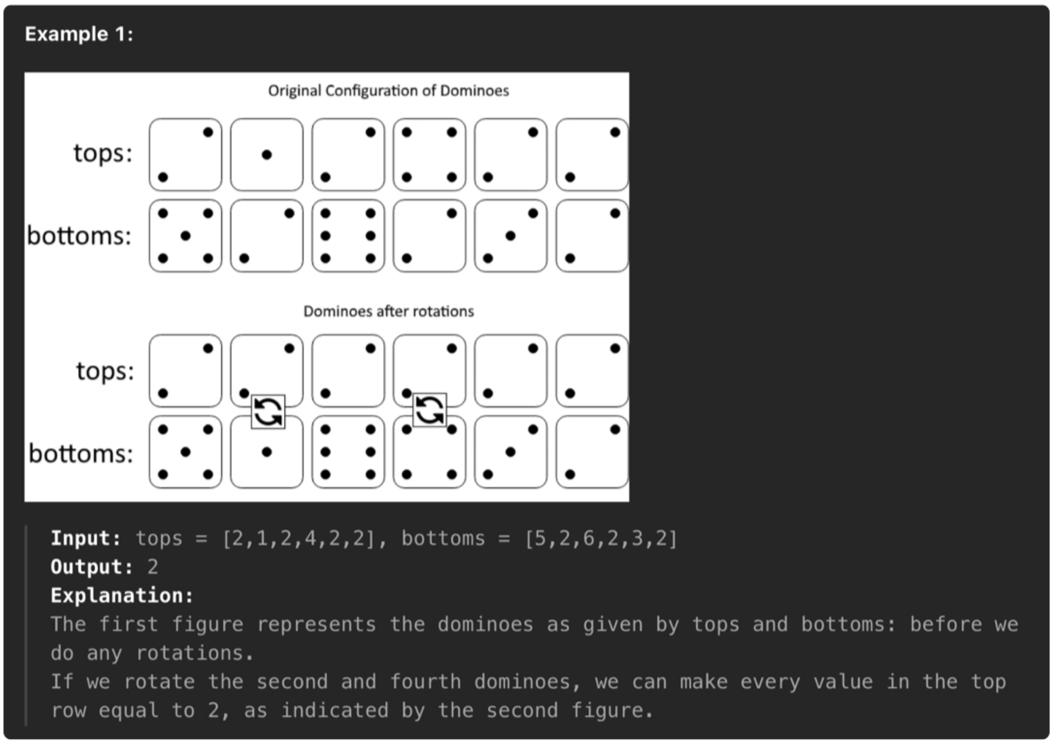
1007. Minimum Domino Rotations For Equal Row

In a row of dominoes, `tops[i]` and `bottoms[i]` represent the top and bottom halves of the `i`th domino. (A domino is a tile with two numbers from 1 to 6 - one on each half of the tile.)

We may rotate the `i`th domino, so that `tops[i]` and `bottoms[i]` swap values.

Return the minimum number of rotations so that all the values in `tops` are the same, or all the values in `bottoms` are the same.

If it cannot be done, return `-1`.



Example 1:

```
Input: tops = [2,1,2,4,2,2], bottoms = [5,2,6,2,3,2]
Output: 2
Explanation:
The first figure represents the dominoes as given by tops and bottoms: before we
do any rotations.
If we rotate the second and fourth dominoes, we can make every value in the top
row equal to 2, as indicated by the second figure.
```

Thought Process

Whenever a problem has a small, limited set of possible values (like 1 to 6), it's often hinting that you can afford brute-force over those values → Greedily.

- Dominoes → 1 - 6
- Dice → 1 – 6
- Cards → 52 cards (13 ranks * 4 suits = 52 Cards)
- Days of weeks, months → 7 days, 12 months etc.
- Directions → E, W, N, S, up, down, left, right
- Vowels → a, e, i, o, u (5 vowels)
- Digit based → 0-9 digits / Pin etc.

```
for( cardValue = 1 ; cardV <=6 ; cardV++) {

        // ~~~~~~

}
```

tops = [2, 1, 2, 4, 2, 2]   topSwap = 1

bottom = [5, 2, 6, 2, 3, 2]   bottomSwap =

i i

⇒ 1 , 2, 3, 4, ⑤, ⑥

```
for( val = 1;  val <= 6 ;  val++) {        O(1)

    int steps    = find (top, bottom, val) ;  ∈ O(n)

      if (steps != -1) {                     O(n)
         result  =  min (result, steps);     S.C = O(1)
      }
}
```

find (top, bottom, val) {

```
// check for top .

for (i = 0;    i < n ;    i++) {        O(n)

      if (top[i] != val && bottom[i] != val) {
                return -1 ;
      }
```

```
        } else if (top[i] != val) {
                swaptop++;
        }
}

// check for bottom

for (i=0;  i<n ;  i++) {        ← O(n)

      if (top[i] != val  &&  bottom[i] != val) {
              return -1 ;
      } else if (bottom[i]  != val) {
              swap bottom ++;
      }
}


return min (swaptop , swapbottom);

}.
```

Check in one for loop:-

```
  ([illegible])  ([illegible])
for (i = 0;    i<n;  i++)  {
```

```
        if (top[i] != val && bottom[i] != val){
                return -1;
        }else if (top[i] != val) {
                swaptop++;
        }else if (bottom[i] != val) {
                swapbottom++;
        }
}

return min(swaptop, swapbottom);
```