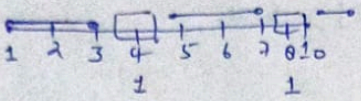
Day_15: DSA

Day-15 Leetcode: 3169 Count Days Without Meetings

input: days: 10 meeting = [[5,7],[1,3],[9,10]]
Output: 2 Range Problem



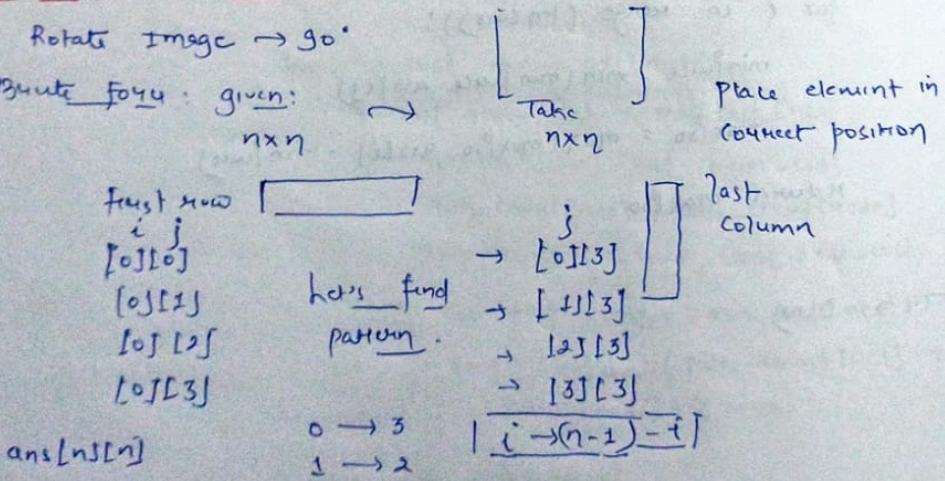
Code:

```
Class Solution:
    def countDays(self, days: int, meetings: List[List[int]]) -> int:
        meetings.sort()
        prev_end = 0
        for start, end in meetings:
            length = end - start + 1
            start = max(start, prev_end + 1)
            days -= max(length, 0)
            prev_end = max(prev_end, end)
        return days
```

• SDE Problem: Rotate Matrix

Rotate Image → 90°

↳ Brute Force: given: n x n → Take n x n → Place element in correct position




```
for (i=0 → n) {
    for (j=0 → n) {
        arr[j][n-1-i] = matrix[i][j]
    }
}
```

T.C → O(N²)
Sc. → O(N²)

• Better Solution:

1st Matrix: → Transpose

2nd → Reverse every Row

```
1  2  3  4          13  9  5  1
5  6  7  8    →     14 10  6  2
9  10 11 12         15 11  7  3
13 14 15 16         16 12  8  4
      ↓ T
1  5  9  13
2  6  10 14
3  7  11 15
4  8  12 16
```

[0][1] → [1][0]
[0][2] → [2][0]
[0][3] → [3][0]
[1][2] → [2][1]

Code:

```
def rotate(matrix : list )
    n= len(matrix)
    # Transpose the matrix
    for i in range(n)
        for j in range(i):
            matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]

    # Reverse each Row for the matrix
    for i in range(n)
        matrix[i].reverse()
```

Problem: Merge Overlapping Subintervals.

given an array of intervals, merge all the overlapping intervals and return an array of non-overlapping intervals

T.C: O(N*logN) + O(2N)
S.C → O(N)

```
def mergeOverlappingIntervals(arr : List[List[int]]) -> int:
    n = len(arr)
    # Sort the given interval
    arr.sort()

    ans=[]
    for i in range(n):
        start, end = arr[i][0], arr[i][1]
        # Skip all the merged intervals
        if ans and end <= ans[-1][1]:
            continue
        # Check for the rest of intervals
```

```
        for j in range(i+1, n):
            if arr[j][0] <= end:
                end = max(end, arr[j][1])
            else:
                break
        ans.append([start, end])
    return ans

# optimal:
    def mergeOver....
        n = len(arr)
        arr.sort()
        ans = []
        for i in range(n):
            # if the current interval does not
            # lie in the last interval
            if not ans or arr[i][0] > ans[-1][1]:
                ans.append(arr[i])
            # if the current interval lies on
                the last interval
            else:
                ans[-1][1] = max(ans[-1][1], arr[i][1])
        return ans.
```