Notes

Date__/__/_____

Day-2

Q: Problem Name: Rotate Image by 90 degree.

1	2	3
4	5	6
7	8	9

⇒

7	4	1
8	5	2
9	6	3

// Brute force:

// Take another matrix (dummy) and then take the first row of matrix and put it in the last col^m, second row → 2nd col^m, and so on....

// TC: O(N * N)

// SC: O(N * N)

Remarks

Date__/__/_____

// Second Approach (optimized approach)

// TC: O(N²)
// SC: O(1)

```
for(int i=0; i<n; i++) {
    for(int j=0; j<i; j++) {
        swap(matrix[i][j], matrix[j][i])
    }
}
for(int i=0; i<n; i++)
    {
        reverse(matrix[i].begin(),
                matrix[i].end());
    }
```

Remarks

Notes

Date__/__/_____

Qus 2

Problem Name : Merge Intervals

greater

smaller

intervals = [[1,3], [2,6], [8,10], [15,18]]

then condition of overlap

[1, 6]

smaller greater

not fulfill condition.

// Brute force : TC (ONlogN + O(N²))
SC : O(N)

// first check array is sorted or not

// Now linearly iterate over the array and check for all next intervals whether they are overlapping or not.

Remarks

// Take a new data structure and insert the overlapped interval.

Note

// optimal approach

// sort intervals

// take two pointers start and end

// iterate over the interval

// if it[0] > end then push in vector
↳ start = it[0]
end = it[1]

// else
end = max(end, it[1])

// push in resultant vector.

TC: O(NlogN + O(
SC: O(N)

Remarks

Notes

Date__/__/_____

Q3 Problem Name:

Merge Sorted Arrays

nums1 = [1,2,3,0,0,0]
nums2 = [2,5,6]
n = 3, m = 3

// Brute force

// We can use a new array of size (n+m) and put all elements of arr1 and arr2 and sort it.

// TC: O(NlogN) + O(N) + O(N)

// SC: O(N)

// optimal approach : TC: O(n+m)
// create a vector SC: O(1)

Remarks

// by comparing num1 and num2, push the num1 and num2 acc to condition in new vector

Date__/__/_____ Not

Problem Name : find duplicate in array of N+1 integers.

nums = [1, 3, 4, 2, 2]
output : 2

// using stl ← approach (1)

// using Binary Search ← approach (2)

// using linked list cycle detection approach (3)

approach 1 : TC ⇒ O(N log N + N)
SC ⇒ O(1)

approach 2 : TC ⇒ O(N)
SC ⇒ O(1)

approach 3 : TC ⇒ O(N)
SC ⇒ O(1)

Remarks