ARRAY [EXTRA CLASS]

Date 12/09/23

Q1. We have an array with both positive and negative numbers. We have to modify the array and keep all -ve numbers in left and +ve in right side.

Approach :-

① Sorting → But takes O(nlogn) T.C

② 2-pointer approach → O(n) T.C

③ Make a temp array and first all -ve numbers in it and the store all +ve ones. → O(n) T.C
↳ Extra, O(n) space complexity.
Storage

↳ Using 2-pointer Approach

Ex:-

	4	2	-8	7	-6	-12	3
arr →							

Logic :-

① 'index' is traversing the array

② 'j' represent the memory block where we can keep -ve numbers

③ arr[index] > 0 → ignore

④ arr[index] < 0 → swap (arr[index], arr[j])
then ↳ j++

Date ..../..../.....

Code :-

```
void swapNegativeOneSide (int arr[], int n){
  int j = 0;
  // j → memory block → where i can store negative numbers

  for( int i=0; i< n; i++){
  // i → looping variable for array traversal
    if( arr[i] < 0) {
        swap (arr[i] , arr[j]);
        j++;
    }
  }
}

int main(){
  int arr[] = { 4, 2, -8, 7, -6, -12, 3};
  int n = 7;
  // function call
  swapNegativeOneSide (arr, n);
  cout << "Printing the array" << endl;
  for( int i=0; i<n; i++){
    cout << arr[i] << " ";
  }
  return 0;
}
```

Output :-

Printing the array

-8 -6 -12 7 2 4 3

Date ..../..../.....

Leetcode Questions

8 Sort Colors

Given an array nums with n objects coloured red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

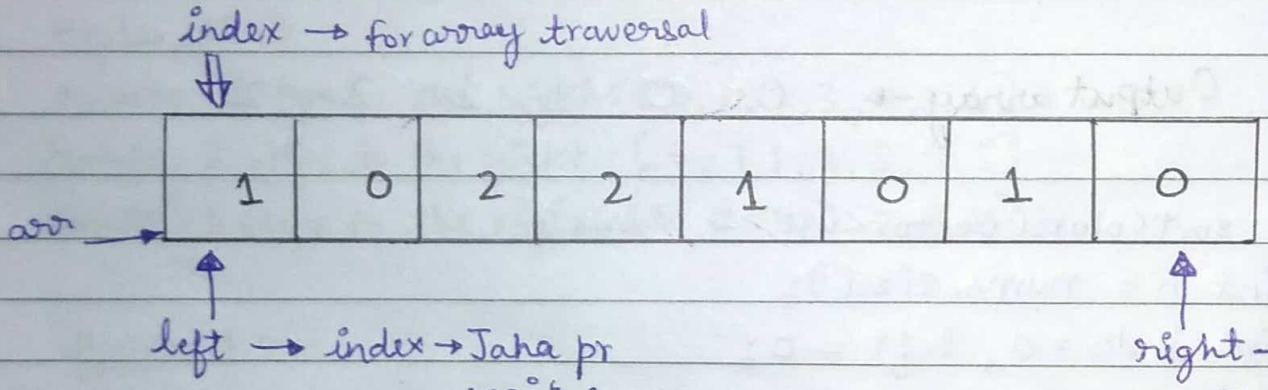
We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:-

Input: nums = [2, 0, 2, 1, 1, 0]

Output: [0, 0, 1, 1, 2, 2]



Logic:-

- '0' mila to left ko dedunga
- 2 mila to right ko dedunga
- 1 mila → ignore and index++

Rule-1:- arr[index] = 1 ✗ (do nothing just → index++)

Rule-2:- arr[index] = 0 → (first) swap → (arr[index], arr[left])

After that → left++ → index++

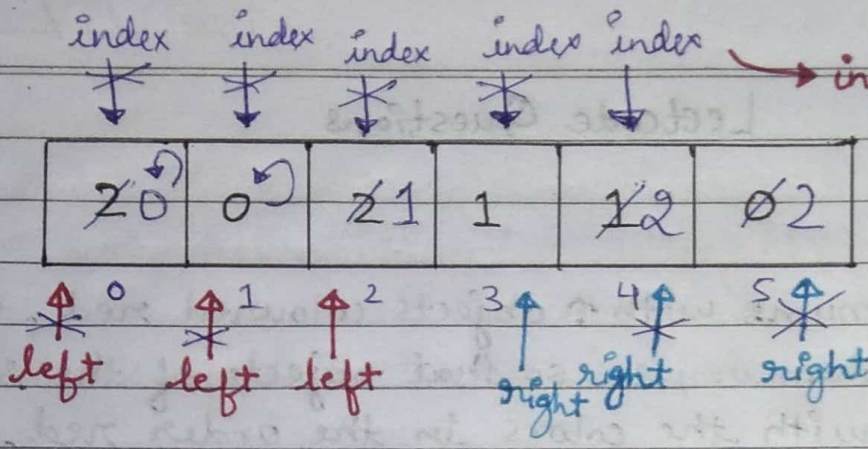
Rule-3:- arr[index] = 2 → swap → (arr[index], arr[right])

→ right-- → index++ ✗ Catch

Yaha mai galti Spiral karunga

Date ..../..../.....

↳ Dry Run :-



→ index > right
↳ stop!
Bcz now we have only 2's so, no need to swap

index=0 → arr[index] = 2 → swap(index, right) → right--
index=0 → arr[index] = 0 → swap(index, left) → left++ → index++
index=1 → arr[index] = 0 → swap(index, left) → left++ → index++
index=2 → arr[index] = 2 → swap(index, right) → right--
index=2 → arr[index] = 1 → index++
index=3 → arr[index] = 1 → index++
index=4
↳ index > right
↳ STOP!

Output array → 0 0 1 1 2 2

Code :-

```
void sortColors(vector<int>& nums){
    int n = nums.size();
    int index=0, left = 0;
    int right=n-1;

    while (index<= right){
        if(nums[index]==0){
            swap(nums[index],nums[left]);
            left++;
            index++;
        }
```

Spiral

Date ..../..../.....

```
else if (nums[index] == 2){
    swap(nums[index], nums[right]);
    right--;
    // catch → no need of index++
}
else{
    index++;
}
}
}
```

Q Rotate Array. (189) Imp ★★★ Amazon

Given an integer array 'nums', rotate the array to the right by 'k' steps, where 'k' is non-negative.

Example 1:-

Input:- nums = [1,2,3,4,5,6,7], K=3

Output:- [5,6,7,1,2,3,4]

Explanation:-

rotate 1 steps to the right: [7,1,2,3,4,5,6]

rotate 2 steps to the right: [6,7,1,2,3,4,5]

rotate 3 steps to the right: [5,6,7,1,2,3,4]

Approach:-

① Modulus

② Temp Array

Input	10	20	30	40	50	60
	0	1	2	3	4	5

Output	50	60	10	20	30	40
	0	1	2	3	4	5

K=2

n=6

∵ Taking extra space to make new vector space

	i/p	o/p
index:	0	2
	1	3
	2	4
	3	5
	4	0
	5	1

index + K ✗

(index + K) % n

(0+2)%6 → 2%6 → 2

(1+2)%6 → 3%6 → 3

(2+2)%6 → 4%6 → 4

(3+2)%6 → 5%6 → 5

(4+2)%6 → 6%6 → 0

Spiral

(5+2)%6 → 7%6 → 1

Date ..../..../.....

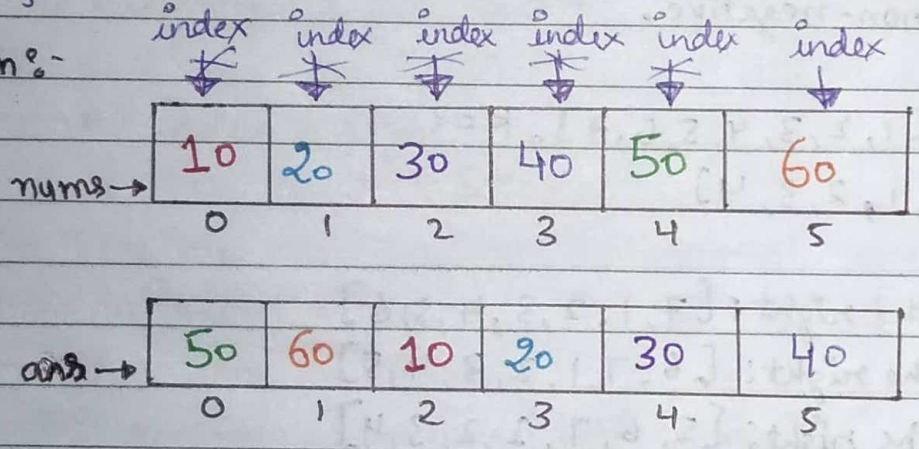
Code :-

```
Void rotate(vector<int>& nums, int K){
    int n = nums.size();
    // Taking additional space to create new shifted array
    vector<int> ans(n);

    for(int index =0; index<n; index++){
        // new index to store the shifted value
        int newIndex = (index + K) % n;
        ans[newIndex] = nums[index];
    }

    nums = ans; // bez nums mein he changes krne hai
}
```

Dry Run :-



K=2, n=6

index=0
newIndex = (0+2)%6 = 2
ans[newIndex] = nums[index]
ans[2] = nums[0]

index = 1
newIndex = (1+2)%6 = 3
ans[newIndex] = nums[index]
ans[3] = nums[1]

index = 2
newIndex = (2+2)%6 = 4
ans[newIndex] = nums[index]
ans[4] = nums[2]

index = 3
newIndex = (3+2)%6 = 5
ans[newIndex] = nums[index]
ans[5] = nums[3]

index = 4
newIndex = (4+2)%6 = 0
ans[newIndex] = nums[index]
ans[0] = nums[4]

index = 5
newIndex = (5+2)%6 = 1
ans[newIndex] = nums[index]
ans[1] = nums[5]

Spiral

Date ..../..../.....

Q Missing Number (268)

Given an array 'nums' containing 'n' distinct numbers in the range [0, n], return the only number in the range that is missing from the array.

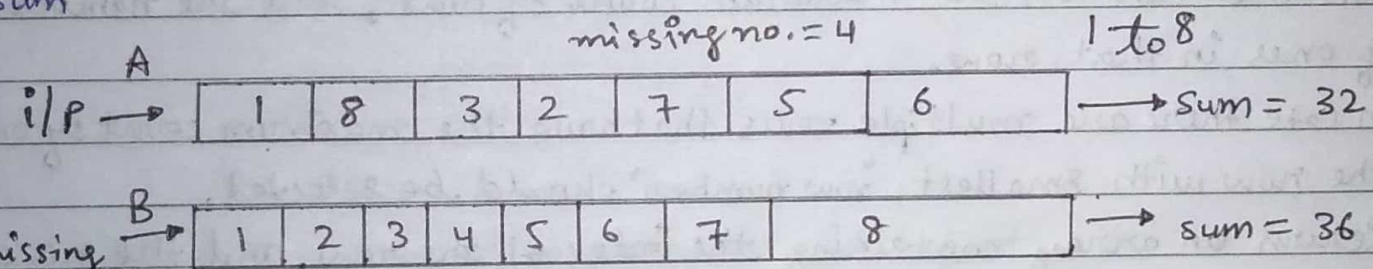
Example 1:- Input : nums = [3, 0, 1]

Output : 2

Explanation:- n = 3 since there are 3 numbers, so all numbers are in the range [0, 3]. 2 is the missing number in the range since it does not appear in nums.

Approaches:-

1. 2pointer → $O(n^2)$ T.C
2. sorting → $O(n \log n)$
   → Traverse & check diff of adjacent no.'s is 1 or not
3. Sum
4. XOR



B - A = 36 - 32

= 4 → missing no found

A.P → $sum = \frac{n}{2}(a + l)$

n → no. of terms

a → 1st term

l → Last term

Date ..../..../.....

Code:-

```
int missingNumber(vector<int>& nums){
    int n = nums.size();
    int sum = 0;
    for(int index=0; index<n; index++){
        sum += nums[index];
    }
    int totalSum = ((n)*(n+1))/2;
    int ans = totalSum - sum;
    return ans;
}
```

Q Row With Maximum Ones (2643)

Given a 'm x n' binary matrix 'mat', find the 0-indexed position of the row that contains the 'maximum' count of 'ones', and the number of ones in that row.

In case there are multiple rows that have the maximum count of ones, the row with smallest 'row number' should be selected.

Return an array containing the index of the row, and the no. of ones in it.

Ex:- Input: mat = [[0,1],[1,0]] Output:- [0,1]

Explanation:- Both rows have the same number of 1's. So we return the index of the smaller row, 0, and the maximum count of ones (1). So, the answer is [0,1].

oneCount = INT_MIN

rowNo = -1

Jab bhi new max oneCount milega, tab row no. store krayenge

	0	1	2	3	
r_0 →	1	0	0	0	→ 1's → 1
r_1 →	0	1	1	0	→ 1's → 2
r_2 →	0	1	1	0	→ 1's → 2
r_3 →	1	1	1	0	→ 1's → 3 → oneCount
r_4 →	0	0	1	0	→ 1's → 1

rowNo

ans = {3, 3}

rowNo → OneCount

Spiral

Date ..../..../.....

Code :-

```
vector<int> rowAndMaximumOnes ( vector< vector<int>>& mat ){
    vector<int> ans;  // array to store answers
    int n = mat.size();
    // oneCount → will store max no. of 1's in a row
    int oneCount = INT_MIN;
    // rowNo → will store row No which contains max no. of 1's
    int row = -1 ;

    for (int i=0;  i< mat.size();  i++) {
    // har row start hone se pehle initialise count with 0
    int count =0;
        for ( int j=0;  j< mat[i].size();  j++ ){
            // If one found, the increment count
            if ( mat[i][j] == 1){
                    count ++;
            }
        }
    // After row completion, compare count with oneCount
    if ( count > oneCount) {
            oneCount = count;
            rowNo = i;
     }
}

ans.push_back( rowNo);
ans. push_back(oneCount);
return 0;
}
```

Date ..../..../.....

Q Rotate Image (48) (Imp) ★★★

You are given an 'n x n' 2D 'matrix' representing an image, rotate the image by 90° (clockwise).

You have to rotate the image in-place which means you have to modify the input 2D matrix directly. DO NOT allocate another 2D matrix and do the rotation.

Example 1:-

1	2	3
4	5	6
7	8	9

⇨

7	4	1
8	5	2
9	6	3

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: [[7,4,1],[8,5,2],[9,6,3]]

2 Step Process to get 90° rotated matrix

↳ 1. Transpose
2. Revers the rows

1	2	3
4	5	6
7	8	9

→ Transpose

1	4	7
2	5	8
3	6	9

↔

7	4	1
8	5	2
9	6	3

Reverse

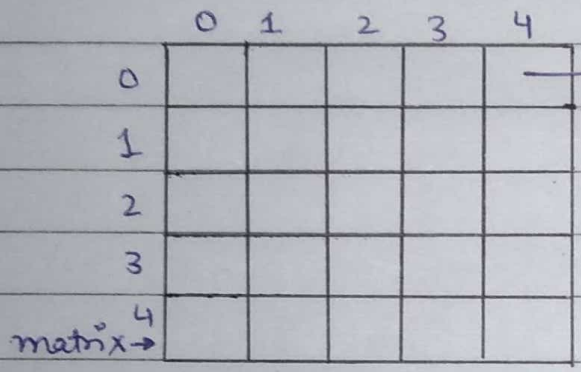
Code:-

```
void rotate(vector<vector<int>>& matrix){
    int n = matrix.size();
    // transpose
    for(int i=0; i<n; i++){
        for(int j=i; j<matrix[i].size(); j++){
            swap(matrix[i][j], matrix[j][i]);
        }
    }
```

Spiral

Date ..../..../.....

```
// reverse → 2D matrix ki sari rows ko
// kitni rows hai → 0 to n-1
for (int i = n-1; i<=0; i--){
    // har row ko reverse krna hai
    reverse (matrix[i].begin(), matrix[i].end());
  }
}
```



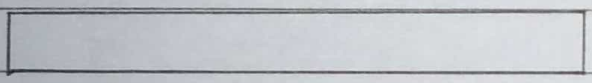
rowName → matrix[0]

→ matrix[1]

i^{th} row → matrix[i]

```
for( i=0 ⟶ i<n)
{
    reverse ⟶ matrix[i]
}
```

↘ 1D Vector<int> arr



reverse (arr.begin(), arr.end())

⟶ reverse (matrix[i].begin(), matrix[i].end())