Arrays - [Extra Class]

Q1. Move Signs.

i/p → {23, 7, 12, -10, -11, 40, 60}

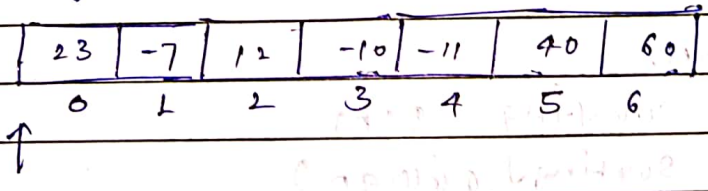
o/p → {-10, -11, 23, 7, 12, 40, 60}

Approaches

(A) Sorting
(B) counting logic
(C) temp Array
(D) Two pointer.

Two pointer approach.

j → for storing negative nos



if arr[index] > 0
↓
ignore

if arr[index] < 0
↳ swap(arr[index], arr[j])
and j++

// Swipper.

```
int j = 0;
for(int i=0; i<n; i++)
{
    if(arr[index] < 0)
    {
        swap(arr[index], arr[j]);
        j++;
    }
}
```

Q. Sort colors.

i/p → | 0 | 1 | 1 | 2 | 0 | 2 | 1 |
arr

o/p → | 0 | 0 | 1 | 1 | 1 | 2 | 2 |

Approaches.

1. Counting O(n)
2. Sorting O(nlogn)
3. Two pointer approach.

| 1 | 0 | 2 | 2 | 1 | 0 | 1 | 0 |

↑ left (for storing 0)

And I will automatically get fixed.

↑ right (for storing 2)

Logic → 0 mila to left ko dedenge.
→ 2 mila to right ko dedenge.
→ 1 automatically beech me aajayega.

Swipper

```
int n = nums.size();
int index = 0;
int left = 0;
int right = n-1;

while (index <= right)
{
    if (nums[index] = 0)
    {
        swap (nums[index], nums[left]);
        left ++;
        index ++;
    }
else if (nums[index] = 2)
    {
        swap (nums[index], nums[right]);
        right --;
    // catch → no need to index ++
    }

else
    {
        index ++;
    }
}
```

Q Rotate Array → by k times.

K = 2

i/p

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

arr

o/p

~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~

50	60	10	20	30	40

↓

Rotated by 2 times.

Approach.

① Module ko use krenge.
② Temp Array.

② Temp Array.

10	20	30	40	50	60

④ 2 size → | 50 | 60 | ← Temp array.

② Modulo wala.

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

i/p

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

Observations

0 → 2 (0+2) %. 6 = 2
1 → 3 (1+2) %. 6 = 3
2 → 4 (2+2) %. 6 = 4
3 → 5 (3+2) %. 6 = 5
4 → 0 (4+2) %. 6 = 0
5 → 1 (5+2) %. 6 = 1

* formula = (index + K) %. n

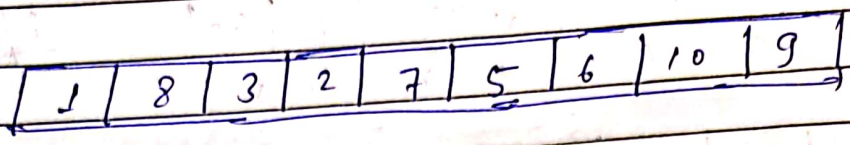
Q. Missing Number

1	7	3	2	5	6	8	10	9

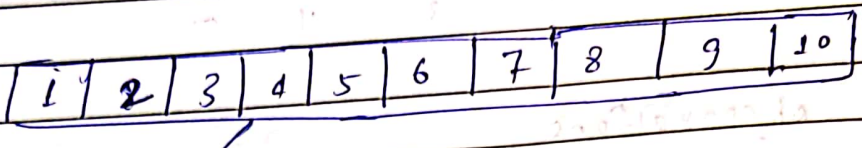
Approach

① find every element in nested loop $O(n^2)$
② sort the array
check the difference between
index numbers. $O(n \log n)$

(3) Sum (AP formula)



→ sum = 32.



→ sum = 36.

missing no.

36 - 32 = 4

// Swipper

```
int sum=0;
int n = nums.size();
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 :

Here we have to find maximum no. of one's row and return the index of that array also..

	0	1	2	3	
0	1	0	0	0	→ 1's → 01
1	0	1	1	0	→ 1's → 02
2	0	1	0	1	→ 1's → 01
3	1	1	1	0	→ 1's → 03
4	0	0	1	0	→ 1's → 01.

Here we have to initialize one count variable for counting and one also for comparing with count variable

And row variable for taking row value.

// swapped

```
vector<int>ans;
int n = mat.size();
// oneCount → will store max nos of 1's in row.
int oneCount = INT_MIN;
//rowNo → will store index of max no of 1's row.
int rowNo = -1;
for(int i=0; i<n; i++){
    int count=0;
    for(int j=0; j<mat[i].size; j++){
        if(mat[i][j]==1) count++;
    }
```

// after row complete, Compare count with
// one count

```
if (count > onecount)
{
    onecount = count;
    rowNo = i;
}
ans.push_back(rowNo);
ans.push_back(onecount);

return ans;
```

Q. Rotate Image. V. imp

- we have to represent a matrix rotated by 90° (in-place)

I/P

1	2	3
4	5	6
7	8	9

→

O/P

7	4	1
8	5	2
9	6	3

Ans Let's see the Transpose of input array

1	4	7
2	5	8
3	6	9

The transpose of o/p matrix seems like the reverse of o/p matrix

So...

The Approach is

1. Transpose the matrix
2. Reverse every rows.

// Snippet

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

↳ you can also make a function to reverse this.

Reverse - Built in function to Reverse any vector or array.

vector → reverse(v.begin(), v.end());

arr → reverse(arr.begin(); arr.end());