Extra CLASS - 12/09/2023
https://www.linkedin.com/in/manojoffcialmj/

ARRAY EXTRA CLASS

1. Moving All Negative Number to the Left Side of an Array

Input

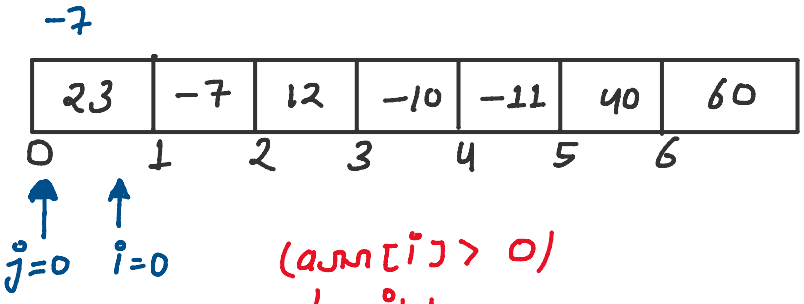
23	-7	12	-10	-11	40	60
0	1	2	3	4	5	6

Output

-7	-10	-11	23	12	40	60
0	1	2	3	4	5	6

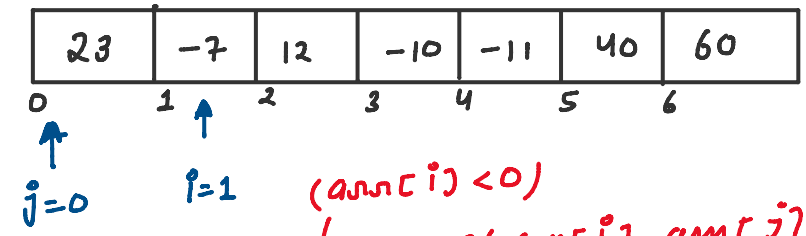
(Two pointers Approach)

Situation: 01



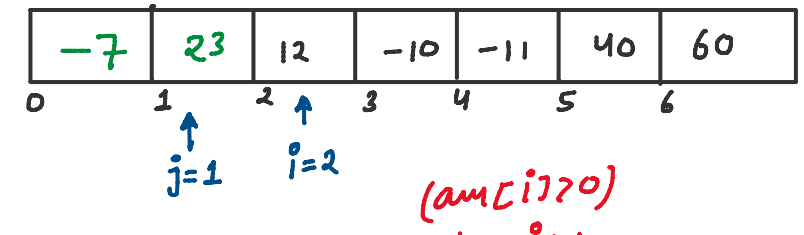
(arr[i] > 0)
↳ i++

Situation: 02



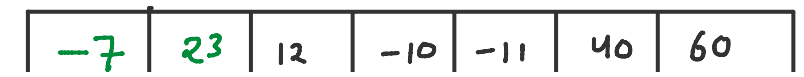
(arr[i] < 0)
↳ swap(arr[i], arr[j]);
i++;
j++;

Situation: 03

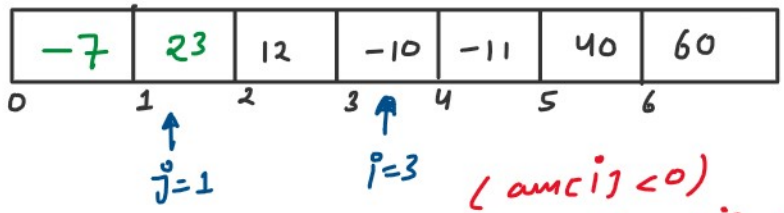


(arr[i] > 0)
↳ i++

Situation: 04

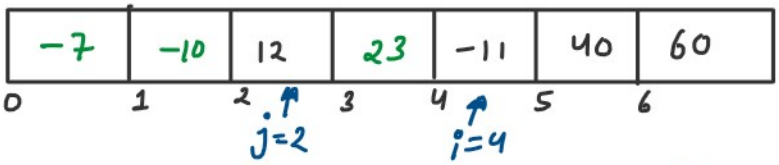


Situation:04



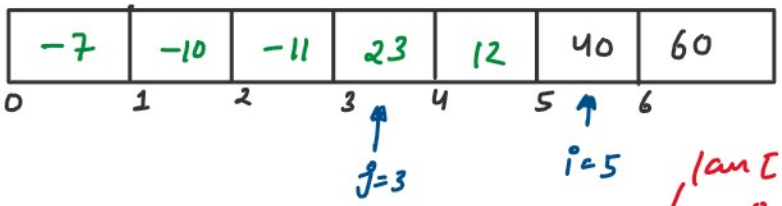
(arr[i]<0)
↳ swap(arr[i], arr[j])
i++
j++

Situation:05



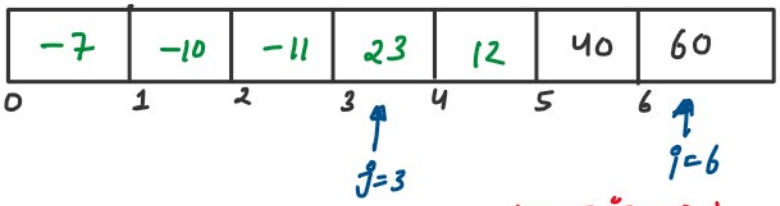
(arr[i]<0)
↳ swap(arr[i], arr[j])
i++
j++

Situation:06



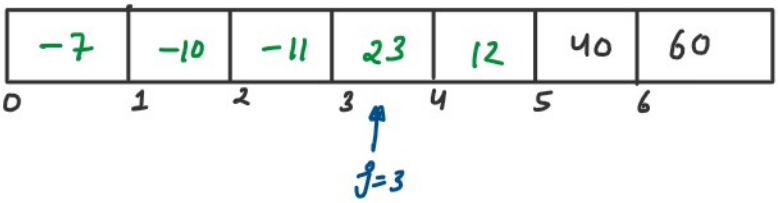
(arr[i]>0)
↳ i++

Situation:07



(arr[i]>0)
↳ i++

Situation:08



i<7 ✗ END

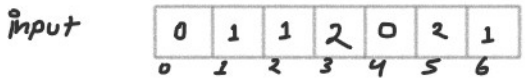
```
// Way 01: Moving All Negative Number to the Left Side of an Array
void shiftNegativeOneSide(int arr[],int size){
    int j=0;
    int i=0;
    while(i<size){
        if(arr[i]>0){
            i++;
```

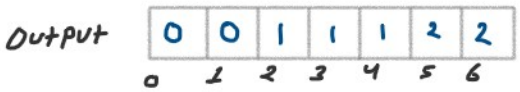
T.C ⇒ O(N)

```
    int i=0;
    while(i<size){
        if(arr[i]>0){
            i++;
        }else{
            swap(arr[i],arr[j]);
            i++;
            j++;
        }
    }
}
// Time Complexity: O(N)
```

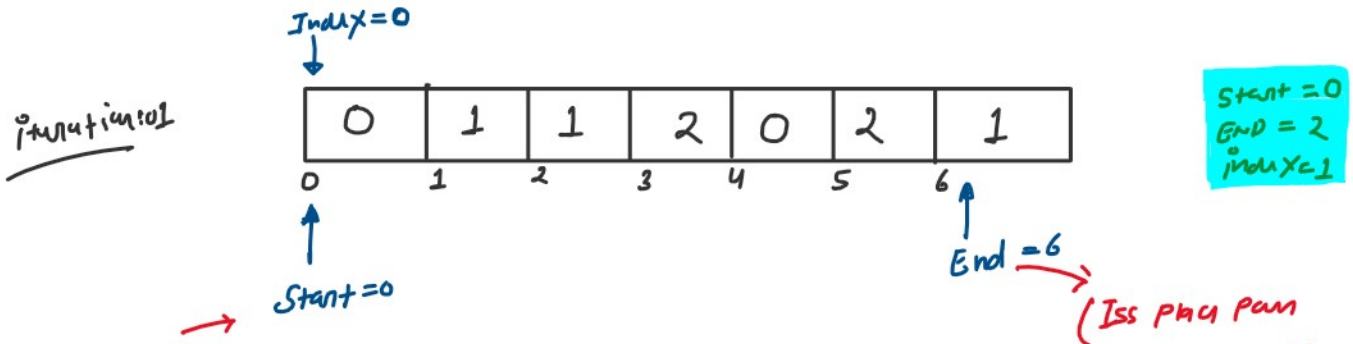
T.C. ⇒ O(N)

2. Sort Colors (Leetcode-75)

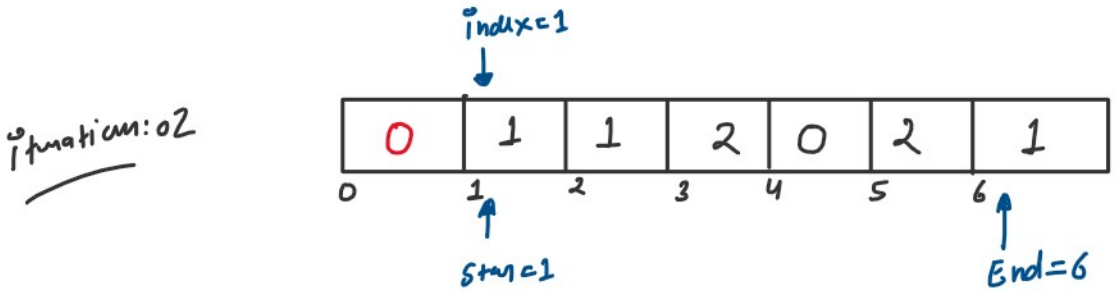




(Two pointers Approach)



(arr[index] == 0)
↳ swap(arr[index], arr[start])
index++
start++

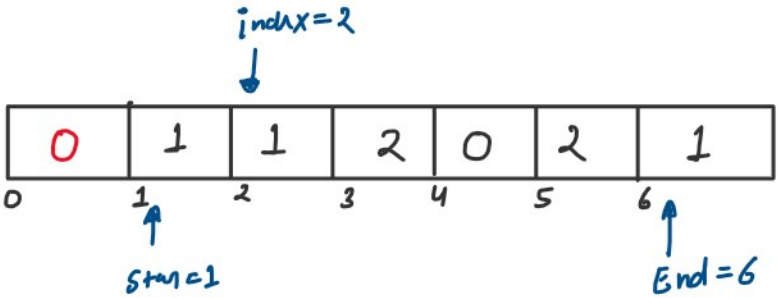


(arr[index] == 1)
↳ index++

(arr[index] == 1)

↳ index++

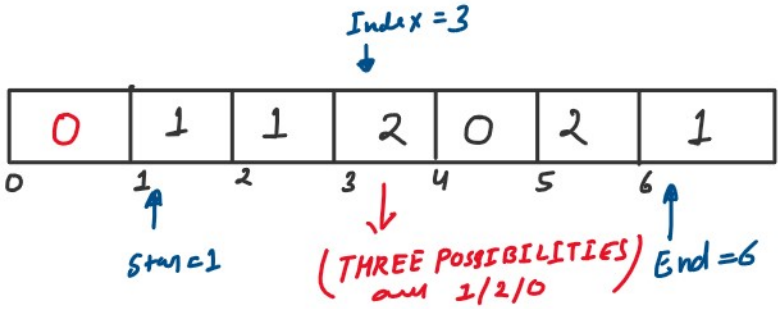
Situation: 03



(arr[index] == 1)

↳ index++

Situation: 04



(arr[index] == 2)

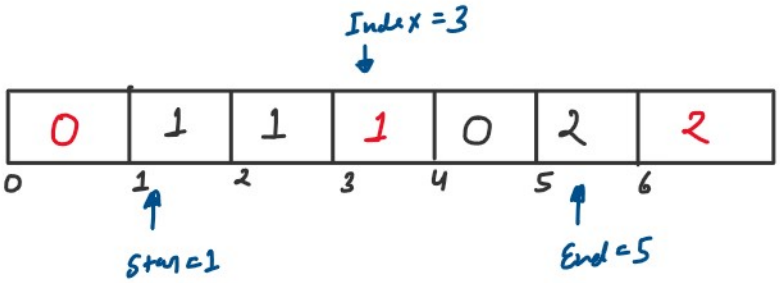
↳ swap(arr[index], arr[End])

End--

index++ X → Yaha par MUJHSE GALTI HO SAKTI HAI

↳ no need to continue

Situation: 05



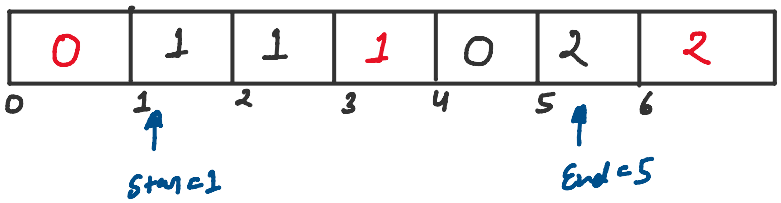
(arr[start] == 1)

↳ index++

index=4

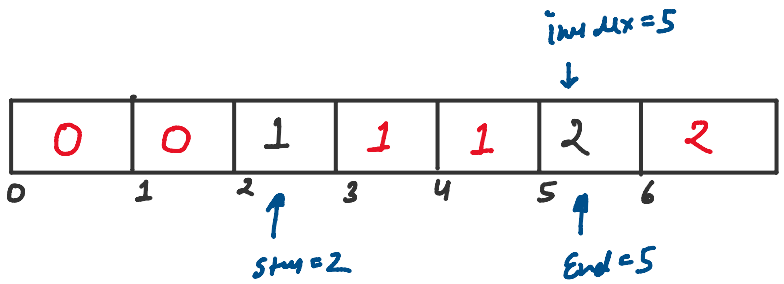
↓

Situation: 06



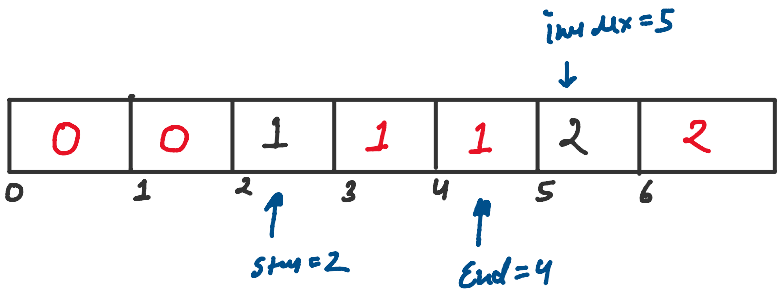
(arr[start]==0)
↳ swap(arr[index], arr[start])
index++
start++

Situation: 07



(arr[start]==2)
↳ swap(arr[index], arr[end])
end--

Situation: 08



(index <= end) ✗ END

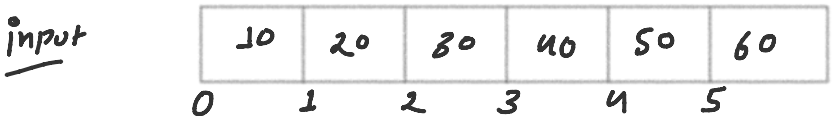
```cpp
// Program 02: Sort Colors (Leetcode-75)

class Solution {
public:
    void sortColors(vector<int>& nums) {
        int start=0;
        int end=nums.size()-1;
        int index=0;
```

T.C. = O(n)

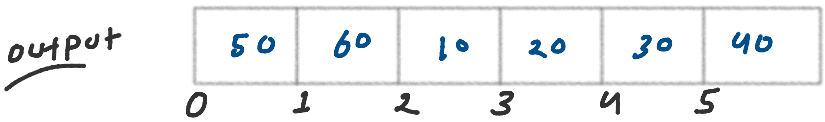
```
        int start=0;
        int end=nums.size()-1;
        int index=0;

        while(index<=end){
            if(nums[index]==0){
                swap(nums[index],nums[start]);
                index++;
                start++;
            }
            else if(nums[index]==1){
                index++;
            }
            else if(nums[index]==2){
                swap(nums[index],nums[end]);
                end--; // Yaha par mujhse galti ho skti hai
            }
        }
    }
};

/*
Example 1:
Input: nums = [2,0,2,1,1,0]
Output: [0,0,1,1,2,2]

Example 2:
Input: nums = [2,0,1]
Output: [0,1,2]
*/
```

T.C. = O(n)

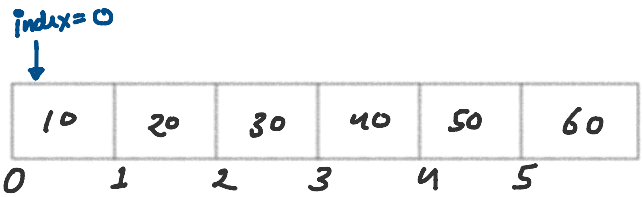
3. Rotate Array (Leetcode-189)



K = 2 , n = 6



Intuition:



① newIndex = (index + K) % n
= (0 + 2) % 6
= 2 % 6

② ans[newIndex] = arr[index]
ans[2] = 10

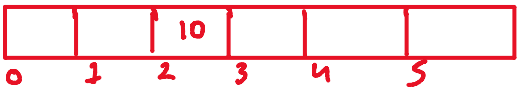
③ index++

= 2 % 6

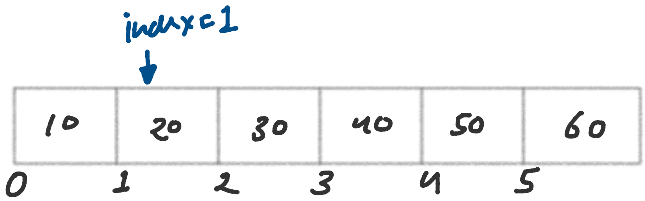
= 2

③ indux++

ANS



Situation:02



① nwIndux = (indux + K) % n

= (1 + 2) % 6

= 3 % 6

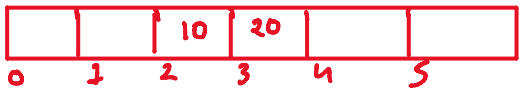
= 3

② Ans[nww[indux]] = ans[indux]

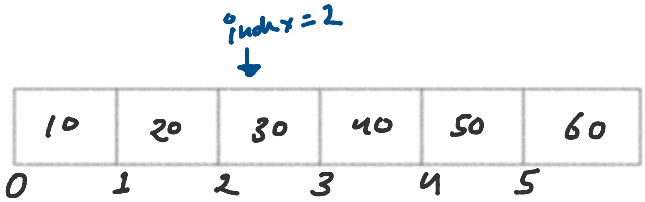
ans[3] = 20

③ indux++

ANS



Situation:03



① nwIndux = (indux + K) % n

= (2 + 2) % 6

= 4 % 6

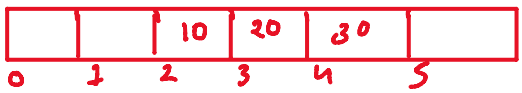
= 4

② Ans[nww[indux]] = ans[indux]

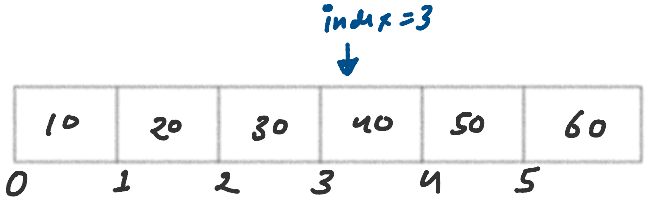
ans[4] = 30

③ indux++

ANS



Situation:04



① nwIndux = (indux + K) % n

= (3 + 2) % 6

② Ans[nww[indux]] = ans[indux]

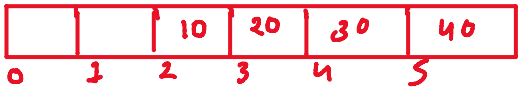
ans[5] = 40

① newIndex = (index + K) % n
= (3 + 2) % 6
= 5 % 6
= 5

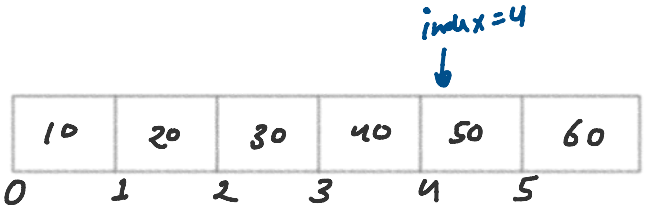
② ans[newIndex] = arr[index]
ans[5] = 40

③ index++

ANS



Situation:05

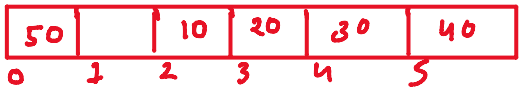


① newIndex = (index + K) % n
= (4 + 2) % 6
= 6 % 6
= 0

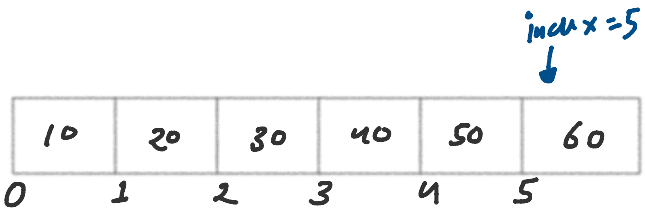
② ans[newIndex] = arr[index]
ans[0] = 50

③ index++

ANS



Situation:06

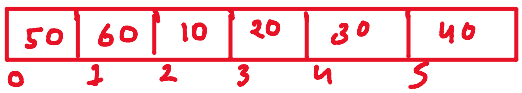


① newIndex = (index + K) % n
= (5 + 2) % 6
= 7 % 6
= 1

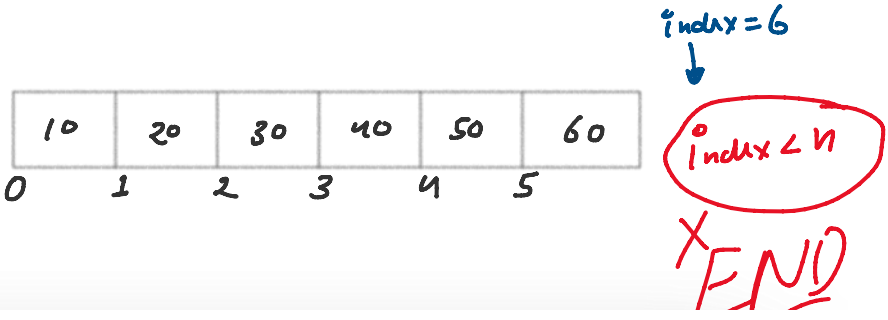
② ans[newIndex] = arr[index]
ans[1] = 60

③ index++

ANS



Situation:07

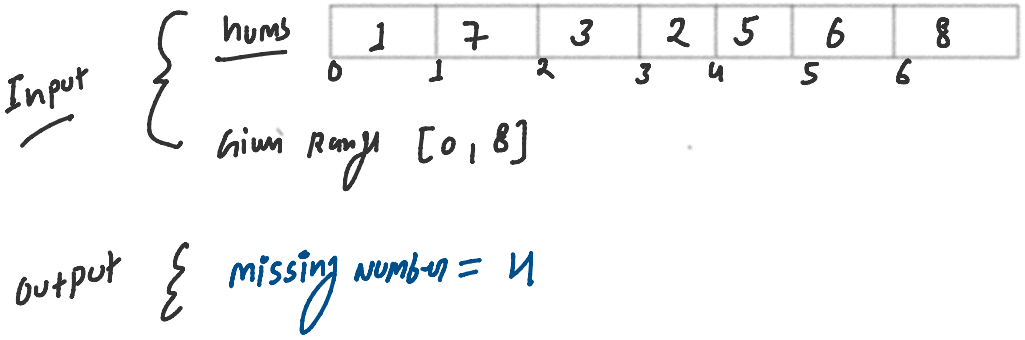


XEND

```cpp
// Program 03: Rotate Array (Leetcode-189)
class Solution {
public:
    void rotate(vector<int>& nums, int k) {
        int n=nums.size();
        vector<int>ans(n);
        int index=0;

        while(index<n){
            int newIndex=(index+k)%n;
            ans[newIndex]=nums[index];
            index++;
        }
        nums=ans; // copy ans into nums because nothing return

    }
};
```

T.C. ⇒ O(n)

4. Missing Number (Leetcode-268)



Approach

step:01 Sum of nums ⇒ 32

step:02 Sum of Range = $n*\left(\frac{n+1}{2}\right)$ ⇒ $8*\left(\frac{8+1}{2}\right) = 4\times 9 = 36$

step:03 missing no = step2 − step1 ⇒ 36 − 32 ⇒ 4

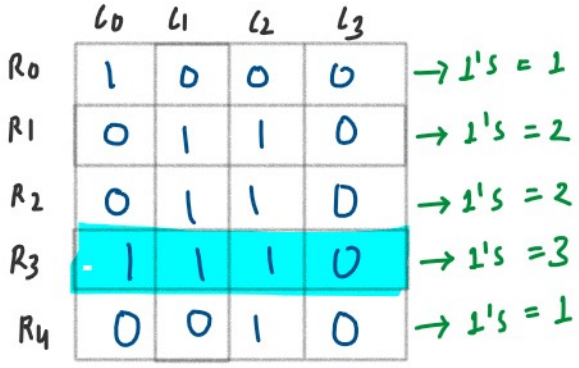
O/P = 4

```cpp
// Program 04:Missing Number (Leetcode-268)
class Solution {
public:
    int missingNumber(vector<int>& nums) {
```

```cpp
// Program 04:Missing Number (Leetcode-268)
class Solution {
public:
    int missingNumber(vector<int>& nums) {
        int n=nums.size();
        int sum = 0;
        for(int i=0;i<n;i++){
            sum+=nums[i];
        }

        int totalSum=(n*(n+1))/2;
        int missingNum=totalSum-sum;

        return missingNum;

    }
};
```

T.C. ⇒ O(N)

5. Row with maximum ones (VVI Leetcode-2643)



MAXIMUM NUMBERS of 1's is 3 at row index 3 so output is

[3, 3]

Row No — One count

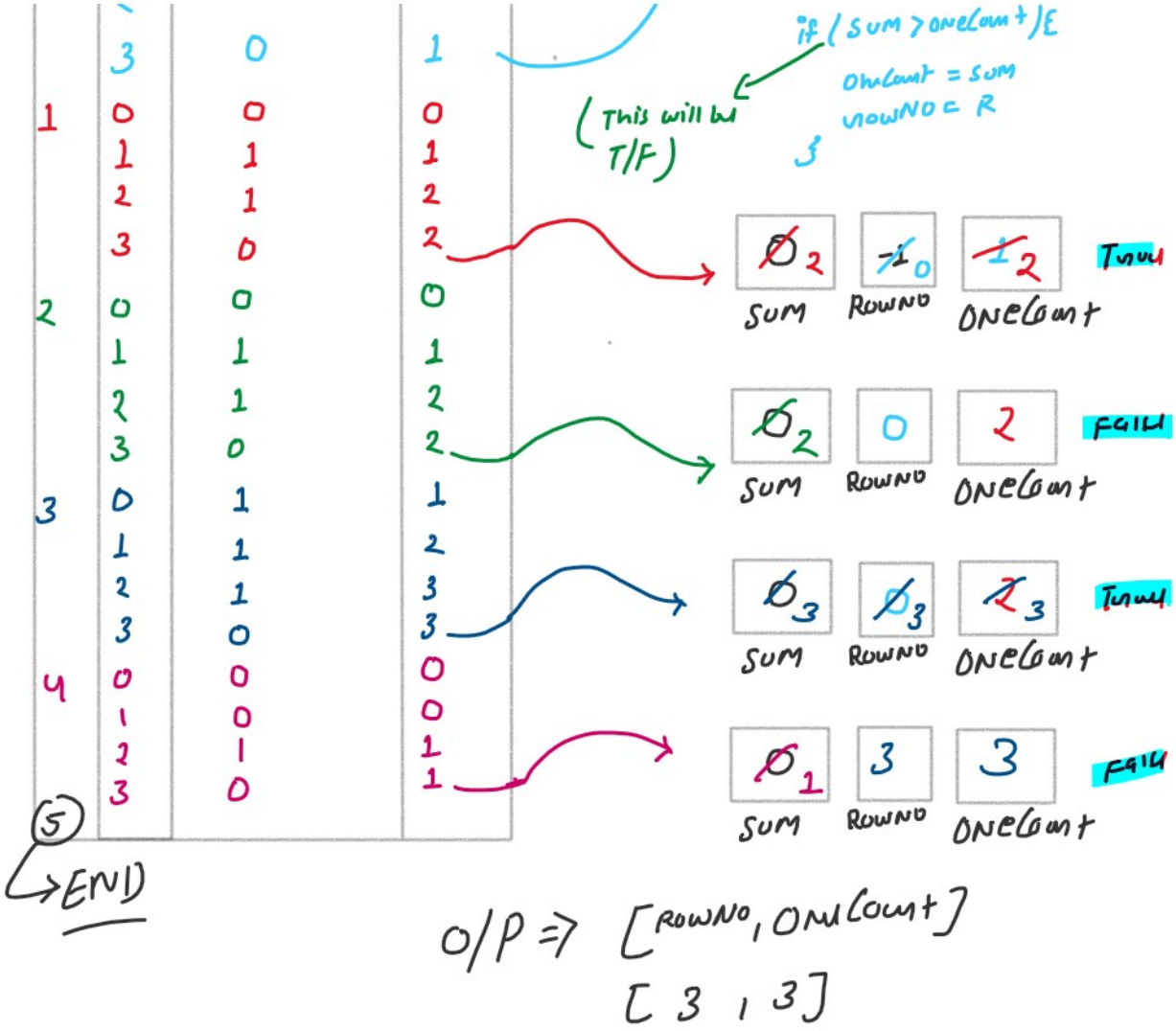
DRY RUN

R	C	MAT[R][C]==1	SUM
0	0	1	1
	1	0	1
	2	0	1
	3	0	1

SUM: ~~0~~ 1 | ROWNO: -1 | ONECOUNT: ~~-1~~ 1

if (SUM > ONECOUNT)

ONECOUNT = SUM



O/P => [ROWNO, ONECount]

[3, 3]

```cpp
// Program 05: Row with maximum ones (VVI Leetcode-2643)
class Solution {
public:
    vector<int> rowAndMaximumOnes(vector<vector<int>>& mat) {
        int rowSize=mat.size();
        int colSize=mat[0].size();

        int oneCount=INT_MIN;
        int rowNo=-1;
        vector<int> ans;

        for(int row=0;row<rowSize;row++){
            int sum=0;
            for(int col=0;col<colSize;col++){
                if(mat[row][col]==1){
                    sum++;
                }
            }

            if(sum>oneCount){
                oneCount=sum;
                rowNo=row;
            }
        }

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

O(n)

O(m)

T.C. => O(n*m)

6. Rotate Image by 90 degree (VVI Leetcode-48)

Input

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

Output

	0	1	2
0	7	4	1
1	8	5	2
2	9	6	3

Approach

STEP:01 Transpose of mat

	0	1	2
0	1	4	7
1	2	5	8
2	3	6	9

STEP:02 Reverse Each Row

	0	1	2
0	7	4	1
1	8	5	2
2	9	6	3

```cpp
// Program 06: Rotate Image by 90 degree (VVImp Leetcode-48)
class Solution {
public:
    void rotate(vector<vector<int>>& matrix) {
        int n=matrix.size();

        // Step 01: Transpose of matrix
        for(int i=0;i<n;i++){
            for(int j=i;j<matrix[0].size();j++){
                swap(matrix[i][j],matrix[j][i]);
            }
        }

        // Step 02: Reverse each from 0 to n-1
        for(int i=0;i<n;i++){
            reverse(matrix[i].begin(),matrix[i].end());
        }

    }
};
```

O(n) O(n) ⇒ O(n*n)

O(n)

= O(n²) + O(n)
= O(n²) T.C.

Note

① int arr[n];

↳ Recursion recursion(arr, arr+n);

② vector<int> arr;

↳ Recursion recursion(arr.begin(), arr.End());

```cpp
// Program 06-1: Rotate Image by 90 degree (VVImp Leetcode-48)
class Solution {
public:
    void reverseVector(vector<int>& arr){
        int size=arr.size();
        int start=0;
        int end=size-1;
        while(start<=end){
            swap(arr[start],arr[end]);
            start++;
            end--;
        }
    }

    void rotate(vector<vector<int>>& matrix) {
        int n=matrix.size();

        // Step 01: Transpose of matrix
        for(int i=0;i<n;i++){
            for(int j=i;j<matrix[0].size();j++){
                swap(matrix[i][j],matrix[j][i]);
            }
        }

        // Step 02: Reverse each from 0 to n-1
        for(int i=0;i<n;i++){
            reverseVector(matrix[i]);
        }

    }
};
```

T.C. $\Rightarrow O(n^2)$