Agenda

1. Single Element 2 (Every element 3 times and one unique element)
2. Single Number 3 (Every element 2 times and 2 unique elements).
3. Given binary array, find number of subarrays having OR 1

< Question > : Given an integer array of size N, where all the elements occur thrice except one element. Find that unique element.

$(1 \le N \le 10^6)$

arr[ ] → [4 , 5 , 5 , 4 , 11 , 6 , 6 , 4 , 5 , 6]
0 1 2 3 4 5 6 7 8 9

ans = 11

BF Idea for every element, iterate on all array elements and find its frequency. If frequency is 1, that element will be the answer.

T.C → $O(N^2)$
S.C → $O(1)$

idea. → Sort the array

[4, 4, 4, 5, 5, 5, 6, 6, 6, 11]

T.C → $O(N \log N)$
S.C → O(depends on sorting algo)

Idea -2

arr[ ] → [5 7 5 4 7 11 11 9 11 7 5 4 4]

0 1 2 3 4 5 6 7 8 9 10 11 12

3 2 1 0

5[ ] →	0	1	0	1
7[ ] →	0	1	1	1
5[ ] →	0	1	0	1
4[ ] →	0	1	0	0
7[ ] →	0	1	1	1
11[ ] →	1	0	1	1
11[ ] →	1	0	1	1
9[ ] →	1	0	0	1
11[ ] →	1	0	1	1
7[ ] →	0	1	1	1
5[ ] →	0	1	0	1
4[ ] →	0	1	0	0
4[ ] →	0	1	0	0
count_set_bit →	4	9	6	10
ans →	1	0	0	1

→ 9

</> Code

```
ans = 0
for( i = 0; i < 32; i++){
    count_set_bits = 0;
    for( j=0; j < N; j++){
        if ((arr[j] & (1<<i)) != 0){
            count_set_bits ++;
        }
    }
    if( count_set_bits % 3 != 0 ){
        //set the ith bit in ans
        ans = (ans | (1<<i));
    }
}
return ans;
```

T.C → O(N)
S.C → O(1)

< *Question* > : Given an integer array of size N, where all elements repeat twice except two. Find those two elements.

arr[ ] → [4 , 5 , 4 , 1 , 6 , 6 , 5 , 2]

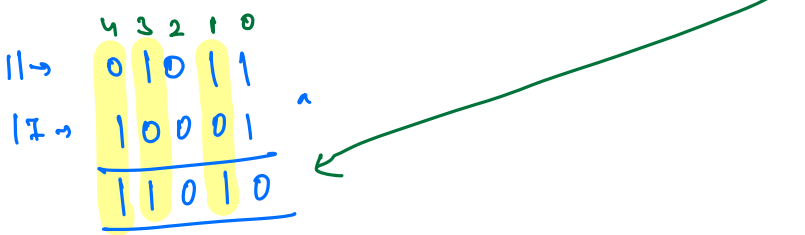
ans → [1, 2]

{ T.C - 0(N)
S.C - 0(1) }

xor → 4^5^4^1^6^6^5^2 → 1^2 → (3)

	0	1	2	3	4	5	6	7	8	9	10	11
arr →	10	8	8	9	12	9	6	11	10	6	12	17
	1010	1000	1000	1001	1100	1001	0110	1011	1010	0110	1100	10001

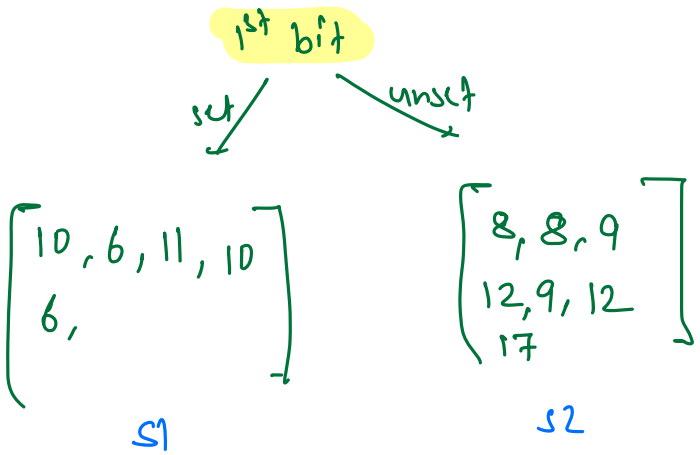
XOR of all the elements → 11^17 → 26



Bits are different at bit position → 1, 3, 4 for the two unique numbers.

⇒ Split all the array elements into two buckets on the basis of 1st, 3rd or 4th bit position →

	0	1	2	3	4	5	6	7	8	9	10	11
arr →	10	8	8	9	12	9	6	11	10	6	12	17
	1010	1000	1000	1001	1100	1001	0110	1011	1010	0110	1100	10001



Step- 1 : Take XOR of all the elements.

```
val = 0 ;
for( i = 0; i < N; i++){
    val = val ^ arr[i] ;
}
```

Step- 2 : Find any set bit position in val.

```
pos = -1 ;
for( i = 0; i < 32; i++){
    if((val & (1<<i)) != 0){
        pos = i , break;
    }
}
```

Step- 3 : Split the array on the basis of posth bit.

```
s1 = 0,  s2 = 0
for( i = 0; i < N; i++){
    if( (arr[i] & (1<<pos)) == 0){
        s1 = (s1 ^ arr[i]);
    }
    else{
        s2 = (s2 ^ arr[i]);
    }
}
```

Step- 4 : Print the unique numbers.

```
print( s1 + " " + s2);
```

T.C → O(N)
S.C → O(1)

Q.

You are given an array consisting of 0s and 1s. Your task is to calculate the number of subarrays for which the bitwise OR of all the elements in the subarray is 0.

arr → [1, 0, 0, 1, 1, 0, 0, 0, 1]
0 1 2 3 4 5 6 7 8

idea. → Find consecutive zeroes and add the no. of subarrays that will be formed from these consecutive 0's in your ans.

code →

```
count = 0;  ans = 0;

for( i = 0; i < n; i++){
    if( arr[i] == 0){
        count++;
    }
    else{
        ans += (count x (count+1)) / 2;
        count = 0;
    }
}
ans += (count x (count+1)) / 2;

return ans;
```

arr[] → [0 0 1 1 0 1 0 0]

count = 1 ✗ 0 ✗ 0 ✗ 2

ans = 3 + 1 + 3 = (7)

< *Question* > : Given binary array, find number of subarrays having OR 1

arr[ ] → [1, 0, 1]

1|0|1 → 1 1|0 → 1 1|0|0|0|0 → 1

The subarrays are: [1], [1, 0], [1, 0, 1], [0], [0, 1], [1].
All subarrays that include the 1 elements have an OR of 1: [1], [1, 0], [1, 0, 1], [0, 1], [1].
ans = 5

Total subarrays = [Subarrays with OR → 1] + [subarrays with OR → 0]

[Total subarrays − subarrays with OR value = 0 = subarrays with OR value → 1]

code →

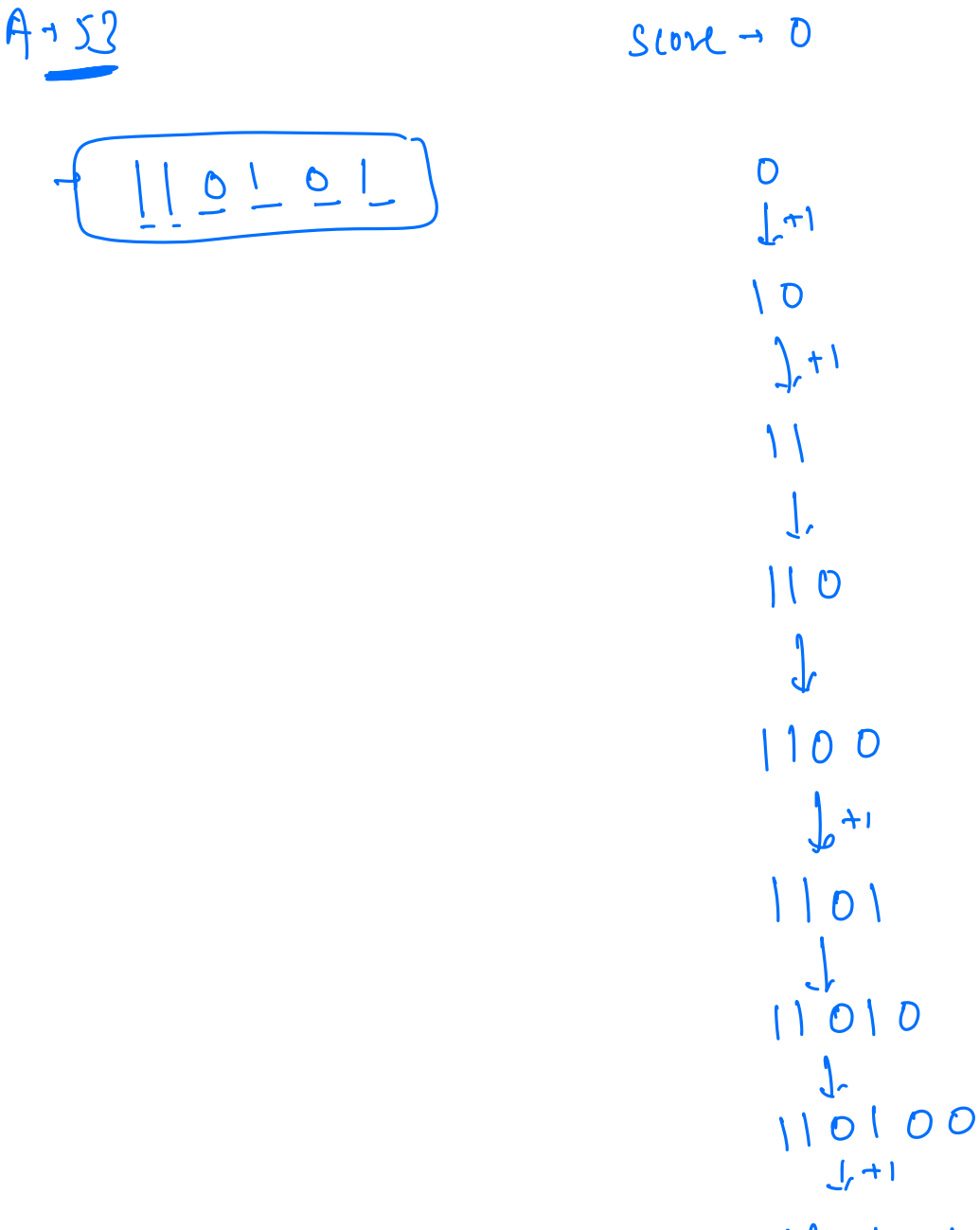
```
count = 0;  ans = 0;

for( i = 0; i < n; i++){
    if( arr[i] == 0){
        count++;
    }
    else{
        ans += (count x (count+1)) / 2;
        count = 0;
    }
}
ans += (count x (count+1)) / 2;
return (n x (n+1))/2 - ans;
```

[TC → O(N)
SC → O(1)]

Alex and Sam are good friends. Alex is doing a lot of programming these days. He has set a target score of A for himself.
Initially, Alex's score was zero. Alex can double his score by doing a question, or Alex can seek help from Sam for doing questions that will contribute 1 to Alex's score. Alex wants his score to be precisely A. Also, he does not want to take much help from Sam.

Find and return the minimum number of times Alex needs to take help from Sam to achieve a score of A.



110101