## Today's Agenda :-

1) Single Number - 1
2) Single Number - 2
3) Single Number - 3
4) Maximum AND pair

Friday : Contest
(Arrays & Bit Manipulations)

Q1) Given an array where every number occurs twice except for one number which occurs just once. Find that no.

Eg arr[]: {4, 5, 5, 4, 1, 6, 6} → ans = 1

Eg arr[]: {7, 5, 5, 1, 7, 6, 1, 6, 4} → ans = 4

Brute force :-

i) Traverse the array & for every element, iterate & get freq : TC : $O(N^2)$

ii) Use Hashmap for storing the freq : TC : $O(N)$
SC : $O(N)$

XOR :- $A \wedge A = 0$ , $A \wedge 0 = A$

$3 \wedge 5 \wedge 3 \wedge 5 = 0$

Since xor helps to cancel out pairs, we can use it.

```
int ans = 0
for(i=0; i<N; i++) {
    ans = ans ^ arr[i]      → xor operation
}
return ans;
```

TC : O(N)
SC : O(1)

Approach 2 : Interesting Solution

Eg arr[] : {2, 3, 5, 6, 3, 6, 2} → ans = 5

| | 2 | 1 | 0 |
|---|---|---|---|
| 2 : | 0 | 1 | 0 |
| 3 : | 0 | 1 | 1 |
| 5 : | 1 | 0 | 1 |
| 6 : | 1 | 1 | 0 |
| 3 : | 0 | 1 | 1 |
| 6 : | 1 | 1 | 0 |
| 2 : | 0 | 1 | 0 |
| | 3 | 6 | 3 |
| | 1 | 0 | 1 |

→ ans = 5

$2x + 1$

```
if(cnt & 1) {
    // the unique ele has a
       set bit at this loc.
}
```

32 bits : [0, 31]

```
int ans = 0

for (int i = 0; i < 32; i++) {
    // count no of set bits at ith pos
    int cnt = 0
    for (int j = 0; j < N; j++) {
        if ((arr[j] & (1 << i)) > 0) {
            cnt++
        }
    }
    if (cnt & 1 == 1) {        // odd check
    if (cnt % 2 == 1)
        ans = ans | (1 << i)
    }
}

return ans
```

TC : O(N)

SC : O(1)

Q2) Given an array, all elements will occur thrice except one element which occurs just once. Find that no.

Eg arr[] : {4, 5, 5, 4, 1, 6, 6, 4, 5, 6} → ans = 1

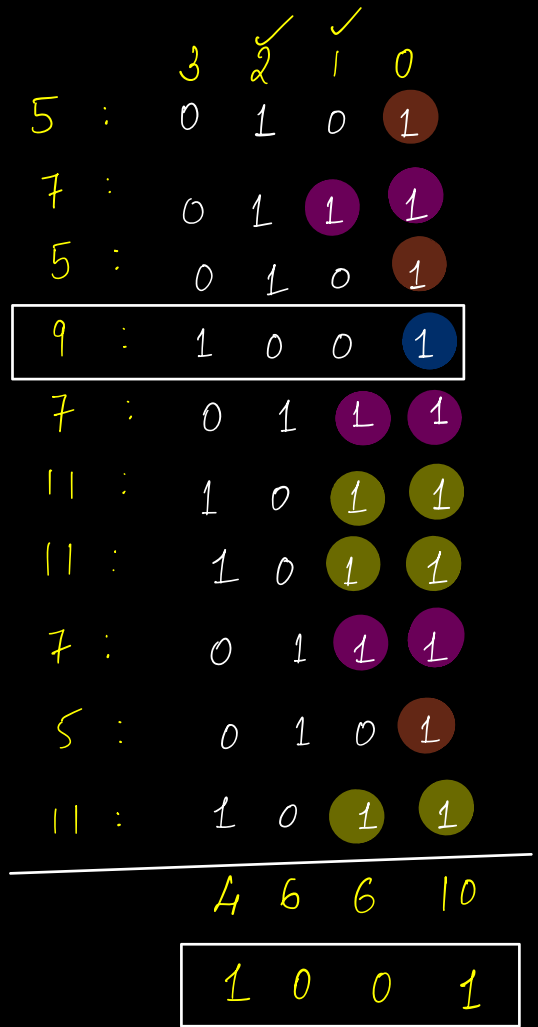
Eg arr[] : {5, 7, 5, 9, 7, 11, 11, 7, 5, 11} → ans = 9

Brute force :-

i) Traverse the array & for every element, iterate & get freq : TC : $O(N^2)$

ii) Use Hashmap for storing the freq : TC : $O(N)$
SC : $O(N)$

arr[] : {5, 7, 5, 9, 7, 11, 11, 7, 5, 11} ⇒ ans = 9 ✓




3 ✓

⇒ ans = 9

```
if(cnt %3 == 1) {
    // there is a set bit
    in ans at this pos
}
```

```
int ans = 0
for(int i = 0; i < 32; i++){
    //count no of set bits at ith pos
    int cnt = 0
    for(int j = 0; j < N; j++){
        if((arr[j] & (1 << i)) > 0){
            cnt++
        }
    }
    if(cnt % 3 == 1){  //if cnt is not a multiple of 3.
        ans = ans | (1 << i)
    }
}
return ans
```

TC : O(N)
SC : O(1)

XOR

Break of 5 Min } After break : 2 more.

Q3) Given an array, all elements will occur twice except 2 elements. Find the 2 elements.

Eg arr[]: {4, 5, 4, 1, 6, 6, 5, 2} → ans = 1 & 2

Idea) XOR of all elements.

{3, 6, 4, 4, 3, 8} = 6 ^ 8

i) XOR of all elements is not directly help.

ii) XOR of all elements = XOR of 2 unique elements.

| | (1000) | (1001) | | (1001) | | (1011) | (0110) | (10001) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| arr[]: | 10 | 8 | 8 | 9 | 12 | 9 | 6 | 11 | 10 | 6 | 12 | 17 |
| | (1010) | | (1000) | | (1100) | | (0110) | | (1010) | | (1100) | |

xor of all elements = 11 ^ 17

| | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| 11 : | 0 | 1 | 0 | 1 | 1 |
| 17 : | 1 | 0 | 0 | 0 | 1 |
| 11^17 : | 1 | 1 | 0 | 1 | 0 |

At bit pos 1, 3, 4, Both unique elements 11 & 17 have different bits.

Bit pos = 1 (Reference)

B1 — elements with bit pos = 1 as Set: {10, 6, 11, 10, 6} → xor of all element = 11

B2 — elements with bit pos = 1 as unset: {8, 8, 9, 12, 9, 12, 17} → xor = 17

Bit pos = 3 (Reference)

B1: {10, 8, 8, 9, 12, 9, 11, 10, 12} → xor ⇒ 11

B2: {6, 6, 17} → xor ⇒ 17

```
xorAll = 0
1) for (int i = 0; i < N; i++) {
       xorAll = xorAll ^ ar[i]
   }

2) // find first set bit in xorAll
   int pos = 0;
   for(int i = 0; i < 32; i++) {
       if ((xorAll & (1 << i)) > 0) {
           pos = i
           break;
       }
   }

3) // divide the array elements keeping pos as a ref.
   int leftXor = 0, rightXor = 0
   for(i = 0; i < N; i++) {
       if ((ar[i] & (1 << pos)) > 0) {
           leftXor = leftXor ^ ar[i]
       }
       else {
           rightXor = rightXor ^ ar[i]
       }
   }

4) 2 unique elements // leftXor & rightXor.
```

TC : O(N)
SC : O(1)

Q4) Given an array, choose 2 indices $(i, j)$ such that $(ar[i] \& ar[j])$ is maximum and $(i \neq j)$. Return the max &.

Eg ar[] : { 5, 4, 6, 8, 5 } (0 1 2 3 4) → ans = 5

i j

0 4 → 5 & 5 = 5

6 : 0110
8 : 1000
6 & 8 : 0000

5 : 101
6 : 110
& : 100 ⇒ 4

ar[] : { 21, 18, 24, 17, 16 }

21 : 1 0 1 0 1 ✓
18 : 1 0 0 1 0
24 : 1 1 0 0 0
17 : 1 0 0 0 1 ✓
16 : 1 0 0 0 0

21 & 17 = 17

ar[] : { 5, 4, 3, 2, 1 } : 5 & 4 =

5 : 101
4 : 100
5 & 4 : 100 = 4

Brute force :- for every pair, get Bitwise AND

TC : $O(N^2)$, SC : $O(1)$

Eg arr[7] : {26, 13, 23, 28, 27, 7, 25}

```
        4  3  2  1  0
26 :    1  1  0  1  0
13 :    0  1  1  0  1
23 :    1  0  1  1  1
28 :    1  1  1  0  0
27 :    1  1  0  1  1
7  :    0  0  1  1  1
25 :    1  1  0  0  1
        >=2 >=2  >=2
```

→ cnt C 2

cnt * (cnt - 1) / 2

ans = 1 1 0 1 0 (Max AND pair)

→ 26

-ve MSB 32 bits

31 ——— 0

MSB

32

31st

```
int ans = 0
for(int i = 30; i >= 0; i--){
    int cnt = 0
    for(int j = 0; j < N; j++){
        if((arr[j] & (1 << i)) > 0){
            cnt++
        }
    }
    if(cnt >= 2){
        // Set ith bit in your ans
        ans = ans | (1 << i);
        for(int j = 0; j < N; j++){
            if((arr[j] & (1 << i)) == 0){
                arr[j] = 0
            }
        }
    }
}
return ans
```

check whether ith bit is set in arr[j]

TC : 31[N+N]

TC : O(N)

SC : O(1)

↳ check whether ith bit is unset in arr[j]

Q5) Calculate the count of pairs for which bitwise & is maximum. (Asked in GOOGLE) → H|W

!

At the end, count no of elements > 0

↓

$cnt\ C_2 = \frac{cnt * (cnt-1)}{2}$ = ans

(5) elements

$5C_2 = \frac{5 \times 4}{2 \times 1} = 10$

$2C_2 = \frac{2 \times 1}{2 \times 1} = 1$

$$^{N}C_2 = \frac{N \times (N-1)}{2}$$

$a_1$
$a_2$
$a_3$
$a_4$
$a_5$
$a_6$

$6C_2 = \frac{6!}{4! \times 2!}$

⇓

$\frac{6 \times 5}{2 \times 1}$

$cnt\ C_2$

$N_{C_r}$ = No of ways of choosing r items out of N

$N_{C_r} = \frac{N!}{(N-r)!\ r!}$

$N_{C_2} = \frac{N(N-1)}{2 \times 1}$

$a_1\ a_2\ a_3 = 3$

$$\frac{3 \times 2}{2 \times 1} = 3$$