SCALER

Bit Manipulation - 2

TABLE OF CONTENTS

1. Single Element - 1
2. Single Element - 2
3. Two Elements
4. Maximum AND Pair

Revision :

AND : $A \& 1$ → 0 (even), 1 (odd)

$A \& 0 = 0$

$A \& A = A$

OR : $A | 0 = A$

$A | A = A$

$A | 1$ → A (odd), A+1 (even)

XOR : $A \wedge A = 0$

$A \wedge 0 = A$

$a << n = a * 2^n$

$a >> n = a / 2^n$

$1 << n = 2^n$

$1 >> n = 1/2^n$

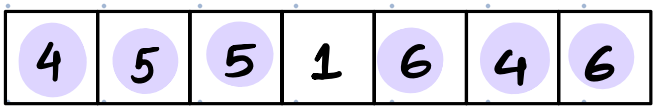
N & (1<<i) : Checks whether i^{th} bit is SET or n't.

N | (1<<i) : Sets i^{th} bit.

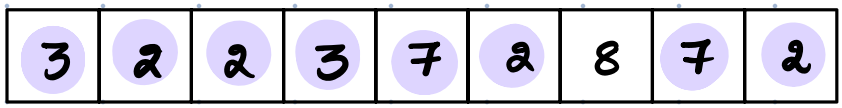
N ^ (1<<i) : Toggles i^{th} bit.

Single Number 1

< Question > : Given arr[N] where every number is present two times except one unique number. Find that unique number.



ans : 1



ans : 8

BF Idea

Traverse the array & find frequency of each element.

```
for (i : 0 ⟶ n) {
    count = 0;
    for (j : 0 ⟶ n) {
        if (A[j] == A[i])
            count++;
    }
    if (count == 1)
        return (A[i]);
}
```

T.C : O(N²)
S.C : O(1)

Idea : Use Hashmap.

T.C : O(N)
S.C : O(N)

Idea : Sort the array.

T.C : O(N log N)
S.C : O(1)

Quiz 1 :

120 ^ 5 ^ 6 ^ 6 ^ 120 ^ 5

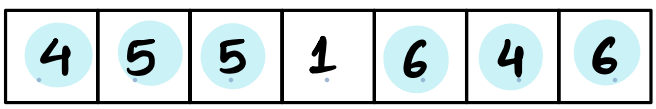
= (120^120) ^ (5^5) ^ (6^6)

= 0 ^ 0 ^ 0 = 0

a^a = 0

Idea 2 :

Take XOR of array



ans : 1

4^5^5^1^6^4^6

= (4^4)^(5^5)^(6^6)^1

= 1

```
int ans = 0;
for (i = 0; i < N; i++) {
    ans = ans ^ A[i];
}
return ans;
```

T.C : O(N)
S.C : O(1)

[Another Approach]

2	2

2 = 0 1 0

2 = 0 1 0

Count of 1's : 0 2 0

Since no appears in PAIRS so count of 1's at each index will be multiple of 2.

2	3	2

2 → 0 1 0

3 → 0 1 1

2 → 0 1 0

count of 1's : 0 3 1

0%2 3%2 1%2

0 1 1 → ans.

Unique no has 1 at the given index if count of 1's is ODD.

Observations :

For every bit position, the count of 1 should be multiple of 2 b'coz numbers appears in pairs, but it can be ODD if the single no has SET BIT at this position.

5 → 1 0 1

3 → 0 1 1

5 → 1 0 1

Count of 1's : 2 1 3

2%2 1%2 3%2

0 1 1

arr[] → | 2 | 3 | 5 | 6 | 3 | 6 | 2 |

ans : 5

	2nd	1st	0th
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
count of 1's :	3	6	3
	3%2	6%2	3%2
	1	0	1

5 → 1 0 1
3 → 0 1 1
5 → 1 0 1
3 → 0 1 1
5 → 1 0 1

count of 1's : 3 2 5

3%2 2%2 5%2

1 0 1

Approach :

count no of set bits at each position

$2x$ → Unique no has '0' at this position

$2x+1$ → Unique no has '1' at this position.

< / > Code

```
ans = 0;
for (i=0; i < 32; i++) {
    count = 0;
    // check ith bit for all numbers
    for (j=0; j<n; j++) {
        if (A[j] & (1<<i) > 0)
            count++;
    }
    // check if count is EVEN or ODD
    if (count & 1 > 0)   // no is ODD
        ans = ans | (1<<i);   // set the ith bit in ans.
}
return ans;
```

T.C : O(N)
S.C : O(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] ans : 11

0 1 2 3 4 5 6 7 8 9

BF Idea

Traverse the array & find frequency of each element

```
for (i : 0 ⟶ n) {
    count = 0;
    for (j : 0 ⟶ n) {
        if (A[j] == A[i])
            count++;
    }
    if (count == 1)
        return (A[i]);
}
```

T.C : $O(N^2)$

S.C : $O(1)$

Idea -2

Sort the array.

2	2	2	3	4	4	4

$a[i] != a[i-1]$ && $a[i] != a[i+1]$

T.C : O(n log n)
S.C : O(1)

Idea : Use Hashmap.

Idea -3

Every element occurs THRICE → count of 1's should be multiple of 3.

If not multiple of 3 → This bit is SET for unique No.

count of set bits at each position

- $3x$ → Unique no has '0' at this position.
- $3x+1$ → Unique no has '1' at this position.

	0	1	2	3	4	5	6	7	8	9	10	11	12
arr[ ] → [	5	7	5	4	7	11	11	9	11	7	5	4	4]

ans : 9

	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 of 1's → 4 9 6 10

4%3 9%3 6%3 10%3

1 0 0 1

ans : 1 0 0 1 → 9

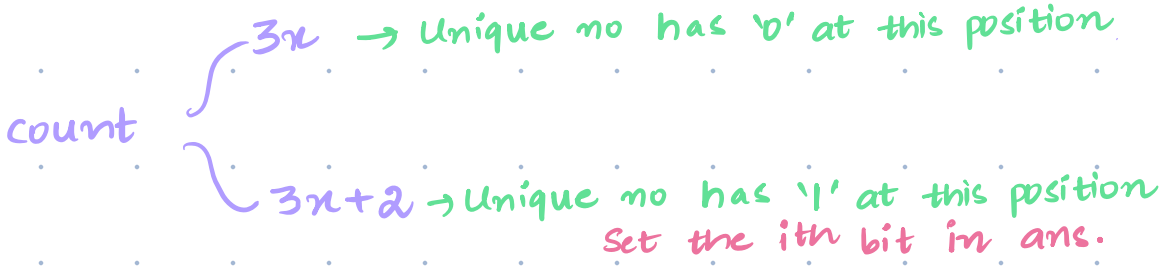
< / > Code

```
ans = 0;
for (i=0; i < 32; i++) {
    count = 0;
    // check ith bit for all numbers
    for (j=0; j<n; j++) {
        if (A[j] & (1<<i) > 0)
            count++;
    }
    // check if count is multiple of 3 or n't
    if (count %3 == 1)
        ans = ans | (1<<i); // set the ith bit in ans.
}
return ans;
```

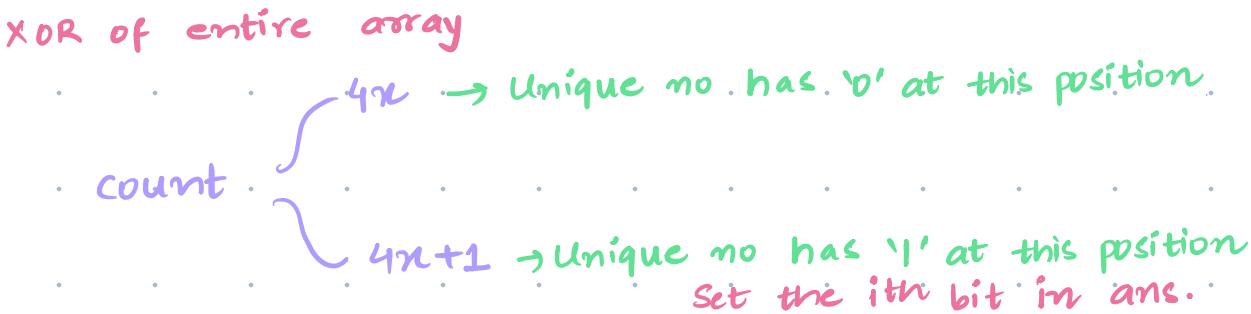
T.C : O(N)
S.C : O(1)

Few Extensions

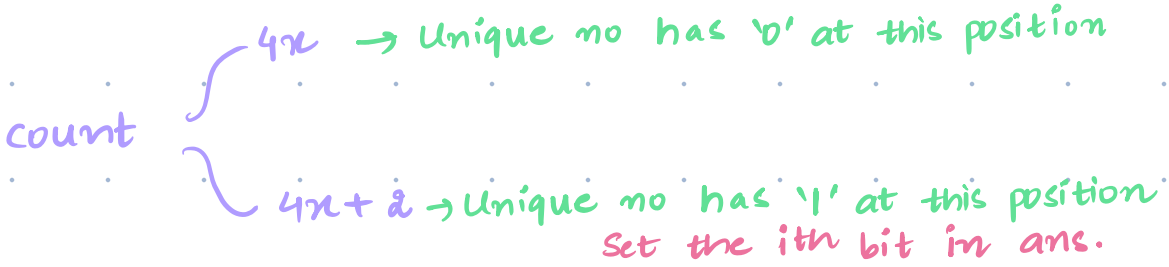
1.Every element is repeating thrice except one, which is repeating twice.



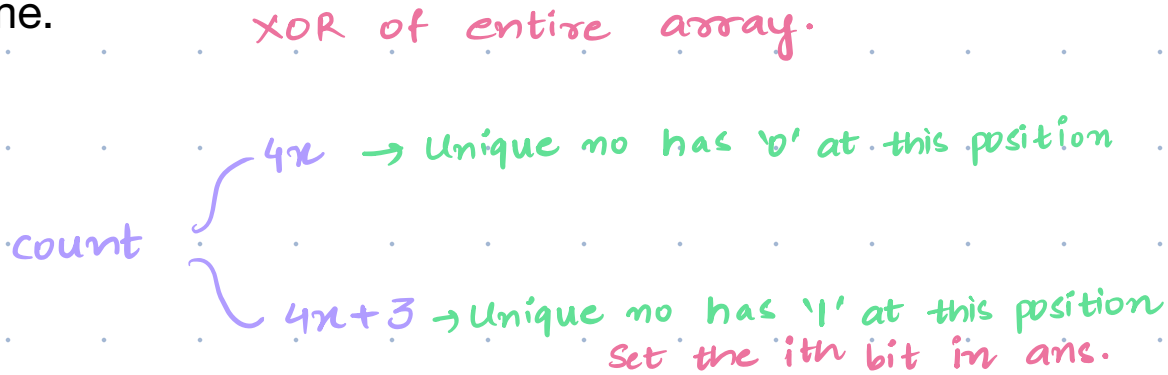
2. Every element is repeating 4 times, except one which is repeating one time.



3. Every element is repeating 4 times, except one which is repeating two time.



4. Every element is repeating 4 times, except one which is repeating three time.



break till 8:25

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

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

ans: {1, 2}

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

BF Idea

Traverse the array & find frequency of all elements and check elements having frequency = 1.

T.C : $O(N^2)$

S.C : O(1)

Idea -2

Sort the array → T.C: (N log N)

Idea -3

XOR of all elements

3	6	4	4	3	8

ans : {6, 8}

3 ^ 6 ^ 4 ^ 4 ^ 3 ^ 8

(3 ^ 3) ^ (4 ^ 4) ^ 6 ^ 8

6 ^ 8

XOR of 2 unique elements

arr →

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

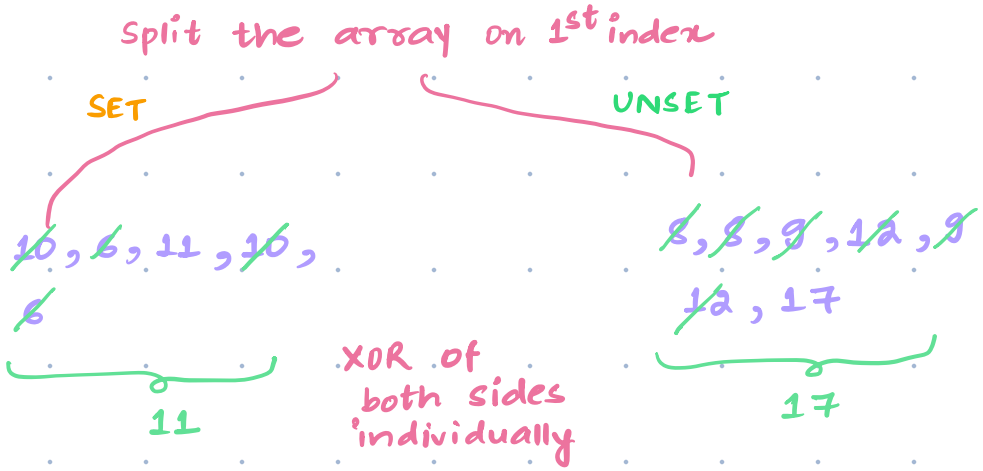
ans : {11, 17}

XOR of all the elements → 11 ^ 17

	4	3	2	1	0
11 =	0	1	0	1	1
^					
17 =	1	0	0	0	1
	1	1	0	1	0

↓ signifies that bits are different for both nos

	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 ^ A[i];
}
```

val = XOR of 2 unique elements

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

```
idx = -1;
for (i=0; i < 32; i++) {
    if (val & (1<<i) > 0)   → ith bit is SET
        idx = i;
        break;
}
```

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

```
set = 0, unset = 0;
for (i=0; i<n; i++) {
    if (A[i] & (1<<idx) > 0)
        set = set ^ A[i];
    else
        unset = unset ^ A[i];
}
```

T.C : O(N)

S.C : O(1)

Step- 4 : Print the unique numbers.

```
print (set);
print (unset);
```

Maximum and Pair

< *Question* > : Given N +ve array elements. Find the maximum value of (arr[i] & arr[j]) where i ! = j (indices must be different for 2 number)

arr[ ] → [27 18 20]

0 1 2

ans : 18

```
27 → 1 1 0 1 1
                &
18 → 1 0 0 1 0
-----------------
     1 0 0 1 0   → 18
-----------------
```

```
27 → 1 1 0 1 1
                &
20 → 1 0 1 0 0
-----------------
     1 0 0 0 0
-----------------
```

```
18 → 1 0 0 1 0
                &
20 → 1 0 1 0 0
-----------------
     1 0 0 0 0
-----------------
```

BF Idea

Consider all possible pairs and find their bitwise AND and max out of it.

```
ans = 0;
for (i : 0 ⟶ n) {
    for (j : i+1 ⟶ n) {
        val = A[i] & A[j];
        ans = max(ans, val);
    }
}
return ans;
```

T.C : $O(N^2)$
S.C : $O(1)$

Optimisation Approach :

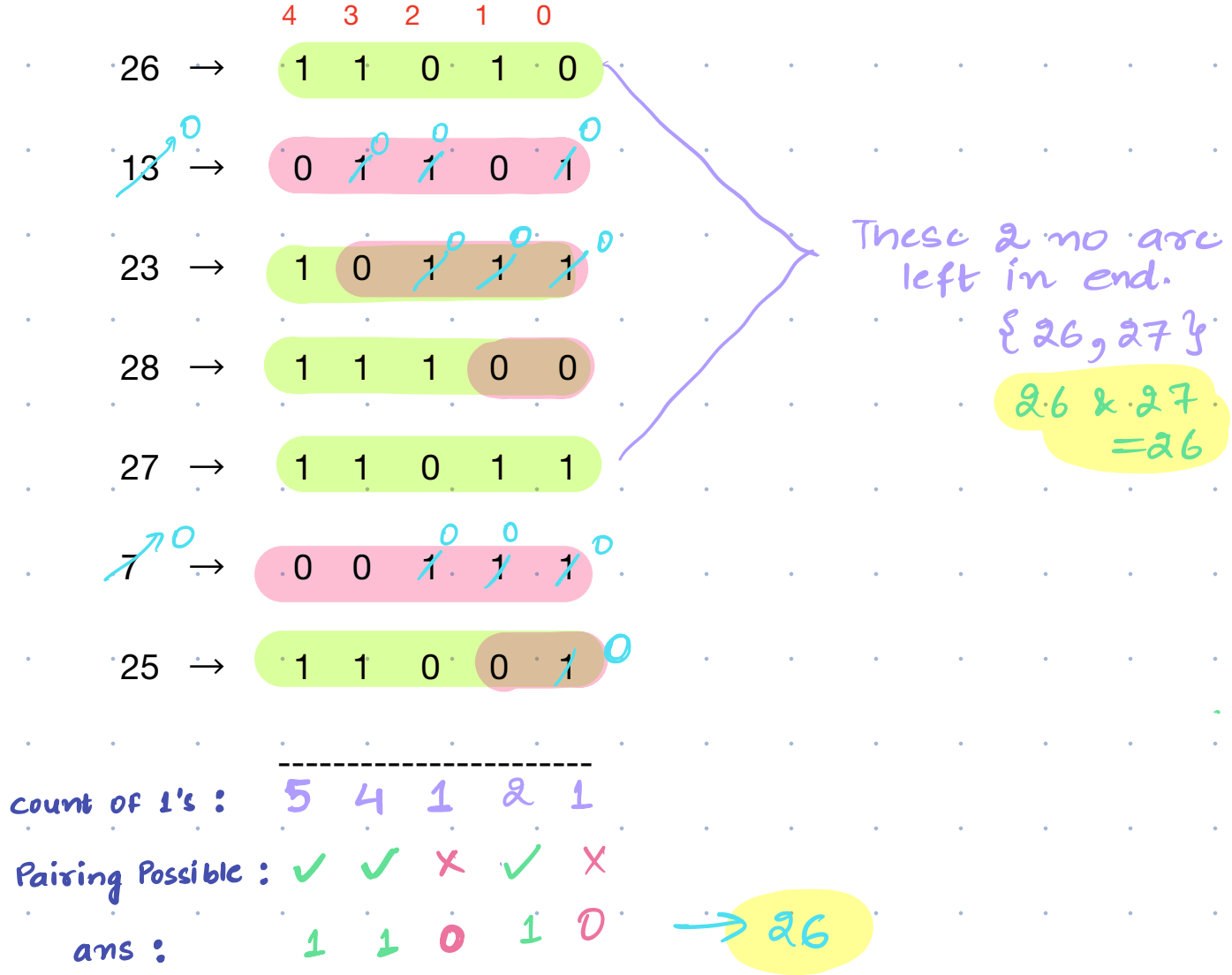
Observations :

- Max value of & : 1
  (when both bits are 1)

- Value of MSB > Value of LSB

  Since MSB contributes more, for ans to be max, we will want set bit to be present towards as left as possible.

arr[ ] → [26 13 23 28 27 7 25] (indices 0 1 2 3 4 5 6)



Approach :

Iterate from MSB → LSB

Count no having set bit at ith position

- >1 → Pairing Possible
  - Set this bit in ans
  - Discard all no having 0 at ith position by making them 0.
- 0/1 → Pairing Not Possible
  - Continue to next bit position.

Quiz 2 :

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

count of 1's : 5 1 1 1 2

Pairing Possible : ✓ X X X ✓

ans : 1 0 0 0 1

ans : 17

{21 & 17}

< / > Code

```
ans = 0;
for (i = 31; i >= 0; i--) {
    // count no having set bits at ith index.
    count = 0;
    for (j = 0; j < n; j++) {
        if (A[j] & (1<<i) > 0)
            count++;
    }

    if (count > 1) { // Pairing Possible
        // Set the ith bit in ans
        ans = ans | (1<<i);

        for (j = 0; j < N; j++) {
            if (A[j] & (1<<i) == 0)
                A[j] = 0;
        }
    }
}
return ans;
```

T.C : O(N)
S.C : O(1)

Question : Find the count of pairs for which bitwise AND is maximum?

After doing above approach, we will see how many pairs are possible from no left in the end.

x No's left.

Pairs : ${}^{x}C_2 = \frac{x(x-1)}{2}$

T.C : O(N)
S.C : O(1)