Lecture : Bit Manipulation-2

Agenda

- Single element
- Single element 2
- Single element 3
- Maximum AND pair
- count of maximum AND pair.

Qu·1 **Single element**

Given arr[n], every element appears twice except one element, find that unique element.

Example

4	5	5	4	1	6	6

7	5	5	1	7	6	1	6	4

Brute force 2 loops

TC: $O(n^2)$

SC: $O(1)$

Approach 2 Using hashmap

TC: $O(n)$

SC: $O(n)$

120 ^ 5 ^ 6 ^ 6 ^ 120 ^ 5 →

$\underbrace{120 \,\hat{}\, 120}_{0} \,\hat{}\, \underbrace{5 \,\hat{}\, 5}_{0} \,\hat{}\, \underbrace{6 \,\hat{}\, 6}_{0} \Rightarrow 0$

Approach3

TC: O(n)

SC: O(1)

4	5	5	4	1	6	6

$4 \wedge 5 \wedge 5 \wedge 4 \wedge 1 \wedge 6 \wedge 6$

$\underbrace{4 \wedge 4}_{0} \wedge \underbrace{5 \wedge 5}_{0} \wedge 1 \wedge \underbrace{6 \wedge 6}_{0}$

$0 \wedge 0 \wedge 1 \wedge 0 \Rightarrow \boxed{1}$ Ans

7	5	5	1	7	6	1	6	4

$7 \wedge 5 \wedge 5 \wedge 1 \wedge 7 \wedge 6 \wedge 1 \wedge 6 \wedge 4$

$\underbrace{7 \wedge 7}_{0} \wedge \underbrace{5 \wedge 5}_{0} \wedge \underbrace{1 \wedge 1}_{0} \wedge \underbrace{6 \wedge 6}_{0} \wedge 4$

$\boxed{4}$ Ans

Code

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

TC: O(n)

SC: O(1)

Qu2

Single element 2

Given arr[n]. Every element appears thrice but there is one element that is unique. find that unique element.

[Amazon
Microsoft
Adobe]

4	5	5	4	1	6	6	4	5	6

Brute force approach

2 loops

TC: O(n²)
SC: O(1)

Approach2

Using hashmap.

TC: O(n)
SC: O(n)

Will previous xor approach work here?

4	5	5	4	1	6	6	4	5	6

$4 \wedge 5 \wedge 5 \wedge 4 \wedge 1 \wedge 6 \wedge 6 \wedge 4 \wedge 5 \wedge 6$

$\underbrace{4 \wedge 4}_{0} \wedge 4 \wedge \underbrace{5 \wedge 5}_{0} \wedge 5 \wedge 1 \wedge \underbrace{6 \wedge 6}_{0} \wedge 6$

$4 \wedge 5 \wedge 1 \wedge 6$ ⟶ not giving me unique el.

Intuition

Case : All no. are coming thrice.

12	8	12	8	12	8

Array el.	Binary representation			
	3	2	1	0
12	1	1	0	0
8	1	0	0	0
12	1	1	0	0
8	1	0	0	0
12	1	1	0	0
8	1	0	0	0
	6 1's 0 0's	3 1's 3 0's	0 1's 6 0's	0 1's 6 0's

No. of 0's and 1's at every index ⇒ multiple of 3.

Our case :- All no. are coming thrice except one number.

12	6	12	3	6	12	6

Array els.	Binary representation			
	3	2	1	0
12	1	1	0	0
6	0	1	1	0
12	1	1	0	0
3	0	0	1	1
6	0	1	1	0
12	1	1	0	0
6	0	1	1	0
	4 0's 3 1's	1 0's 6 1's	3 0's 4 1's	6 0's 1 1's

0 0 1 1 =3

At every bit idx —

0's → multiple of 3

1's → wont be multiple of 3

1

1's → multiple of 3

0's → wont be multiple of 3

0

Extend: Given arr[n]. Every element appears 4|5|6|7 ... n times but there is one element that is unique find that unique element.

Code

```
int singleElement2 (int[] A) {
        int unique = 0;                      int has 32 bits
        for( i = 31; i>=0; i--) {
            int ones = 0;
            for(int el: arr) {
                int bitValue = el & (1<<i);  ≃ 1<<k
                if (bitValue != 0) {
                        ones++;
                }
            }
            if(ones % 3 == 0) {
                    Do nothing
            } else {
                unique += Math.pow(2, i);
                                 1<<i
            }
        }
    return unique;
}
```

TC: O(n)

SC: O(1)

Qu3 Single element 3

Given arr[n], two integers appear only once and all other integers appear twice. find two integers that appears once. (Google)

1	2	3	1	2	4

{ 3, 4 }

1	2

{1, 2}

Prerequisite :- Right most set bit [RSB] mask.

57 :- 1 1 1 0 0 1 ⟶ 0 0 0 0 0 1

76 :- 1 0 0 1 1 0 0 ⟶ 0 0 0 0 1 0 0

2's complement.

RSB ⇒ x & x''

Dry run :- 72 :- 1 0 0 1 0 0 0 $\xrightarrow{RSB}$ 0 0 0 1 0 0 0

```
                          1 1 1
Ones complement :-  0 1 1 0 1 1 1
                +               1
                -----------------
x''  <------        0 1 1 1 0 0 0
&                   1 0 0 1 0 0 0
                -----------------
                    0 0 0 1 0 0 0  ==> RSB.
                -----------------
```

Approach:

36	50	24	56	36	24	42	50

Array el.	Binary representation.
36	1 0 0 1 0 0
50	1 1 0 0 1 0
24	0 1 1 0 0 0
56	1 1 1 0 0 0
36	1 0 0 1 0 0
24	0 1 1 0 0 0
42	1 0 1 0 1 0
50	1 1 0 0 1 0

Step 1: XOR.

36 ^ 50 ^ 24 ^ 56 ^ 36 ^ 24 ^ 42 ^ 50

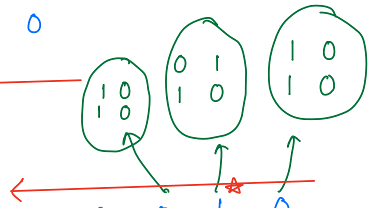
36 ^ 36 ^ 24 ^ 24 ^ 56 ^ 42 ^ 50 ^ 50

$\boxed{42 \wedge 56}$

Step 2:

```
   42 :      1  0  1  0  1  0
^  56 :   ^  1  1  1  0  0  0
          ---------------------
             0  1  0  0  1  0
          ---------------------
```

xor of whole array ⇒



0 1 0 0 1 0

RSB ⇒ 000010

5	4	3	2	1	0
0	0	0	0	1	0

Array el.	Binary representation.					
	5	4	3	2	1	0
36	1	0	0	1	0	0
50	1	1	0	0	1	0
24	0	1	1	0	0	0
56	1	1	1	0	0	0
36	1	0	0	1	0	0
24	0	1	1	0	0	0
42	1	0	1	0	1	0
50	1	1	0	0	1	0

At 1'st idx —

Bitvalue = 0 — el & rsbm=0

36, 24, 56, 36, 24

xor

36 ^ 36 ^ 24 ^ 24 ^ 56

$\boxed{56}$

Bitvalue = 1 — el & rsbm != 0

50, 42, 50

xor

50 ^ 50 ^ 42

$\boxed{42}$

Code

```
void uniqueElement3(int[] A){
    int xor = 0;
    for(int el: A){          —— o(n)
        xor = xor ^ el;
    }
    int rsbm = xor & twocomplement(xor);     h|w

    int x=0;
    int y=0;
    for(int el: A){ —— o(n)
        if(el & rsbm ==0) {
            x = x^el;
        } else{
            y = y^el;
        }
    }
    print(x);
    print(y);
}
```

TC: O(n)

SC: O(1)

Break: 8:36 – 8:47

Qu: Maximum and pair ☆☆☆☆

Given arr[n], choose two indices (i,j) such that (i != j) and arr[i] & arr[j] is maximum.

5	4	6	8	5

⟶ 5 & 5 = 5

21	18	24	17	16

⇓

✓ a.) 21 & 17 :

21 : 1 0 1 0 1

17 : & 1 0 0 0 1

1 0 0 0 1 = 17

b.) 24 & 21

24 : 1 1 0 0 0

21 : 1 0 1 0 1

1 0 0 0 0 = 16

⋮

c.) 17 & 16

d.) 24 & 18

5	4	3	2	1

5 & 4 max ans

Approach

26	13	23	28	27	7	25

Binary representation. ← indices

Array el.	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

Idx ⇒ 4

set bit count = ⑤ 1 _ _ _ _

Binary representation. ← indices

Array el.	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

Idx=3 set bit count = 4 1 1 _ _ _

Binary representation. ← indices

Array el.	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

Idx2:

	Binary representation.					
Array el.	4	3	2	1	0	← indices
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	

set bit count = 1

1 1 0 ___ ___

Idx:1

	Binary representation.					
Array el.	4	3	2	1	0	← indices
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	

set bit count = 2

1 1 0 1 ___

Idx0:

	Binary representation.					
Array el.	4	3	2	1	0	← indices
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	

set bit count = 1

$\boxed{1\ 1\ 0\ 1\ 0}$

↓

26 Ans.

Code:

```
                int  maxAndPair(int[] A) {

                      int  ans = 0;

starting from ——      for(i= 31 ;  i>=0;  i--)  {
MSB to LSB
                            int cnt =0;

                            for(int el: arr) {

count set bits  <——             if(el & 1<<i != 0) {

                                      cnt++;
                                }
                            }
                            if(count >=2)  {

                                ans =  ans + Math.pow(2,i);
                                             1<<i

                                for(j=0; j<arr.length; j++){

ignorance  <——                      if(arr[j] & 1<<i ==0){

                                          arr[j]=0;

                                    }
                                }
                            }
                      }
                      return  ans;
                }
```

TC: O(n)

SC: O(1)

Qus5 Count of pairs with maximum AND [Google] H/w

Calculate no. of pairs for which bitwise AND is maximum.

Do exactly as above and then traverse the array

↓

cnt = count el != 0

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

TC: O(n)

SC: O(1)

Thankyou ☺

Contest: 7 : 8:30

Discussion: 8:30 — ∞

2	4	3	2	3	2	3

~

ones = 0

two = 0

ones = ones ^ (2 &