Single Number I

Single Number II

Single Number III

Max AND Pair

Max AND Pairs

1. We are given an integer array where every no. occurs twice except for one number which occurs just once. Find that number.

A[] = 4, 5, 5, 4, 1, 6, 6 ans = 1

A[] = 7, 5, 5, 1, 7, 6, 1, 6, 4 ans = 4

BF → Go to every element, calculate freq of element (by traversing array)

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

Outer loop → to fix an el
Inner loop → to count freq

Approach 2 : TC : O(N) SC : O(N)

A[] = 4, 5, 5, 4, 1, 6, 6

Build hashmap <elem, freq>

ele : freq
4 : 2
1 : 1
5 : 2
6 : 2

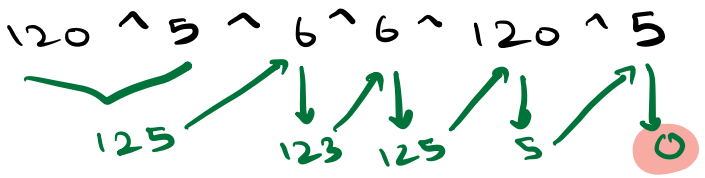
① Traverse array and store freq

② Iterate on hashmap and return key (elem) where value (freq) = 1

Approach 3: TC : O(N) SC : O(1)

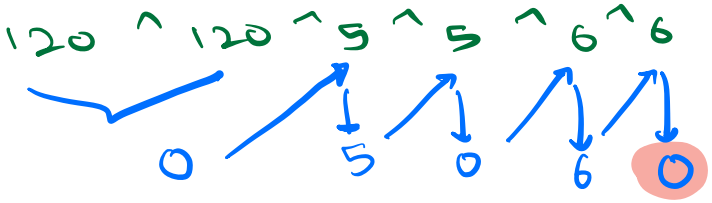
Idea : $A \wedge A = 0$

XOR → pairs cancel out each other



$A \wedge A = 0$

$0 \wedge A = A$



(5 + 10) + 7

15 + 7 = 22

(7 + 5) + 10

12 + 10 = 22

5, 5, 6, 6, 5, 6

xor = 0

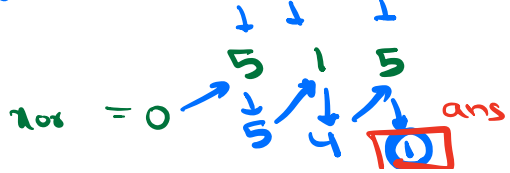
5, 5, 0

xor = 0

```
int xor = 0
for (i = 0 ; i < n ; i++) {
    xor = xor ^ arr[i]
}
return xor
```

TC : O(N)

SC : O(1)



every dc → 4 times

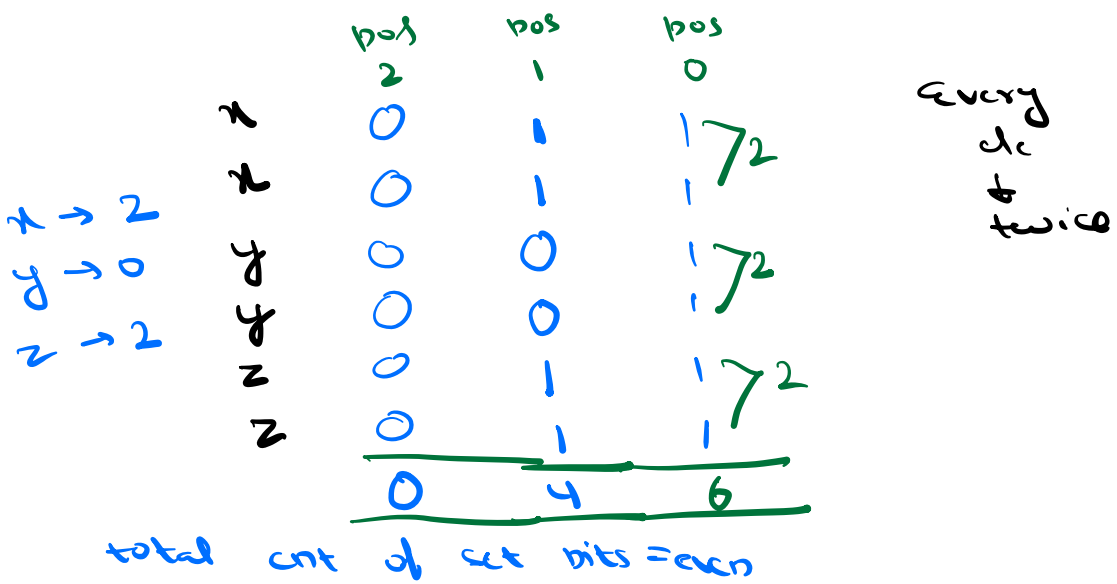
unique dc → 1 time

Delete → Even

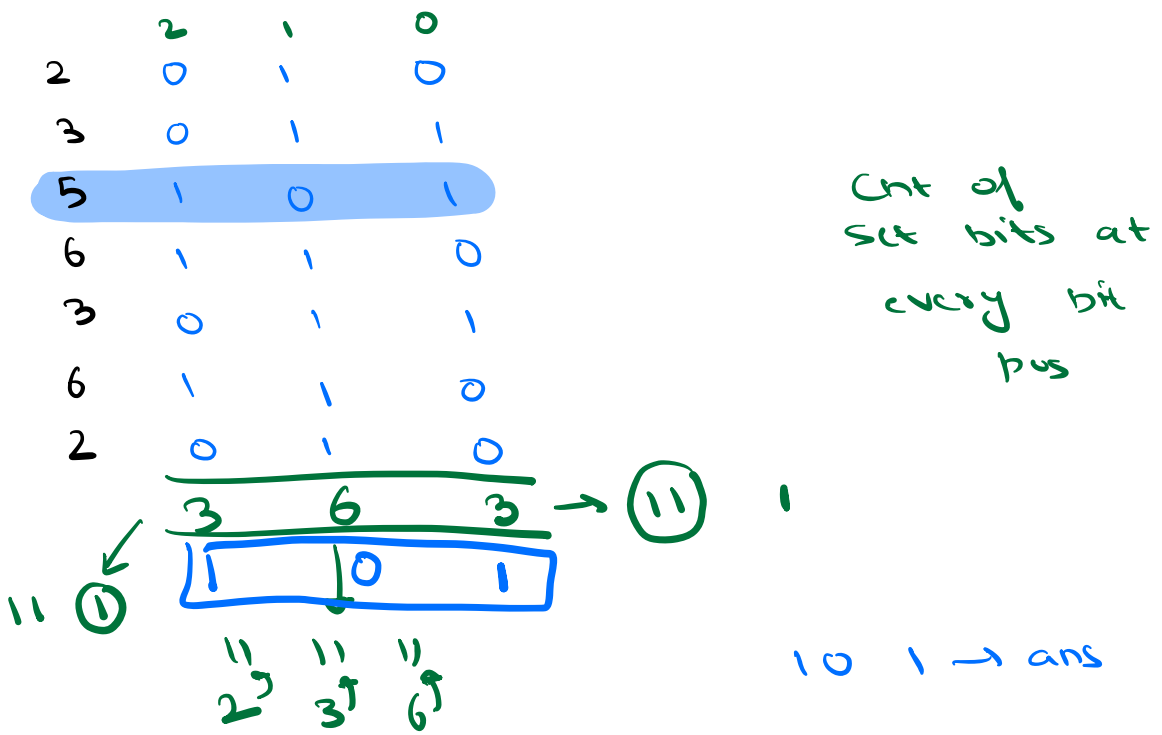
ans → odd

Approach 4:

ans = 5

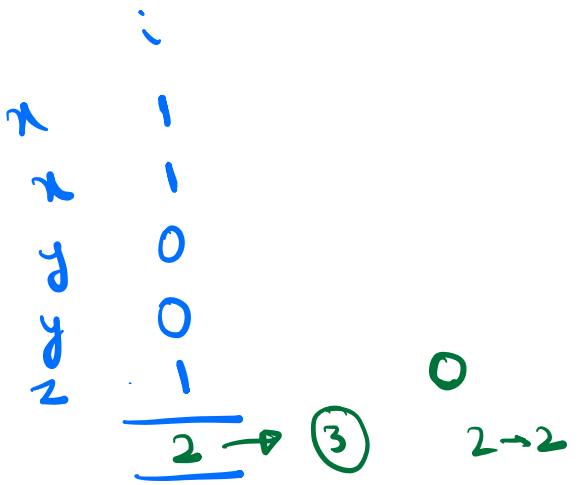


A[] = 2, 3, 5, 6, 3, 6, 2



Cnt of set bits at every bit pos

10 1 → ans

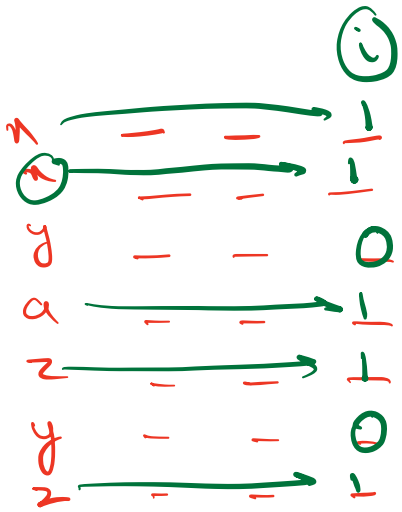


Cnt of set bits at (at pos x) → even / odd

Unique no.

unset at pos x bit

set bit at pos x



5 → 11 11 1

1

```
int ans = 0                                   // arr of integers
for (int i = 0 ; i < 32 ; i++) {
                                              TC : O(N)
    int cnt = 0                               SC : O(1)
    // cnt set bits at ith bit pos

    for (int j = 0 ; j < n ; j++) {
        if (arr[j] & (1<<i) != 0)
            cnt++
    }
    if (cnt % 2 == 1) {
        ans = ans | (1<<i)        → set ith bit in ans
    }
}
print (ans)
```

set a bit in bit pos

5 4 3 2 1 0

ans =

cnt

odd even

0 0 0 0 0 1

---

2. Given integer array, all elements occur thrice except one. Find the unique element.

A[ ] = 4, 5, 5, 4, 1, 6, 6, 4, 5, 6

ans = 1

occurs only once

BF → 1 loop → to fix el

2 loop → traverse ar and count freq

TC : O(N²)

SC : O(1)

Approach 2 - Store freq of every elem in HM

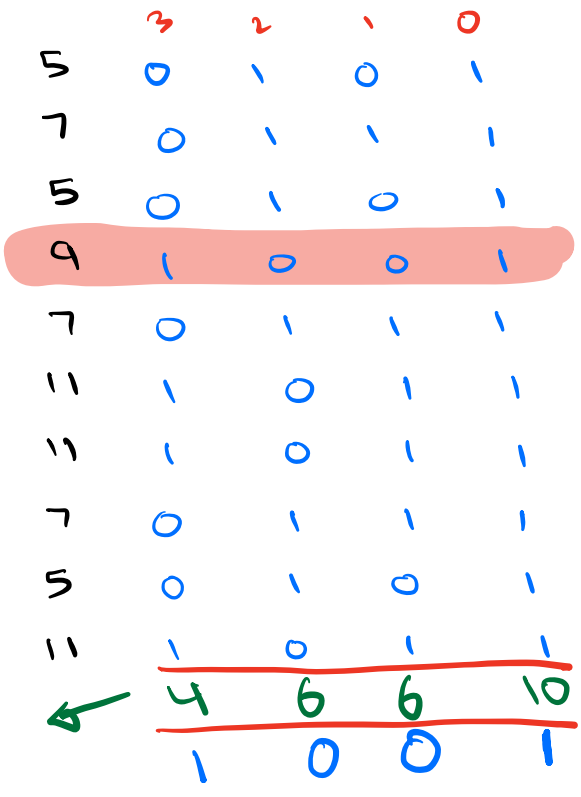
TC : O(N)

SC : O(N)

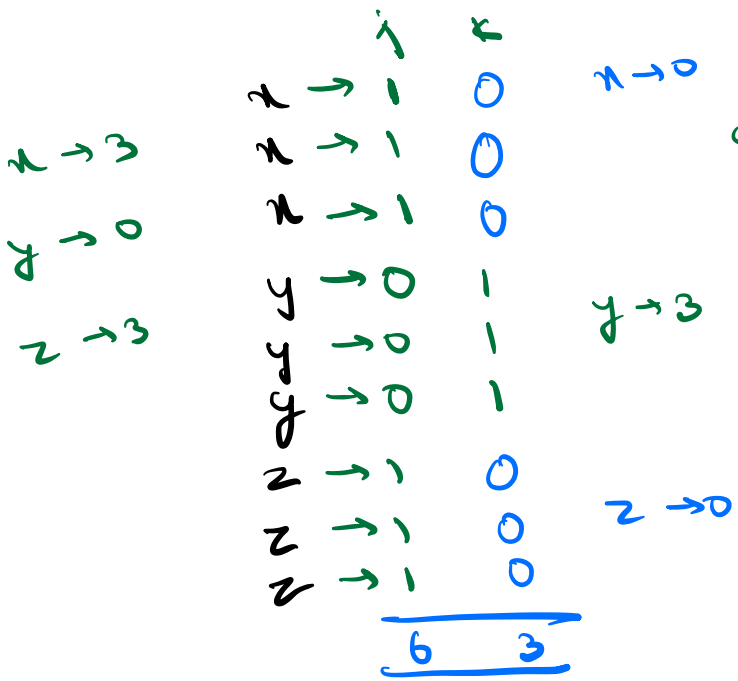
Approach 3

ans = 9

A[] = 5, 7, 5, 9, 7, 11, 11, 7, 5, 11



ans = 1 0 0 1



cnt of set bits will be multiple of 3

cnt set bits for every bit pos

cnt set bits at bos n

cnt % 3 = 1 → set bit at bos n in unique dem

cnt % 3 = 0 → unset bit at bos n in unique de

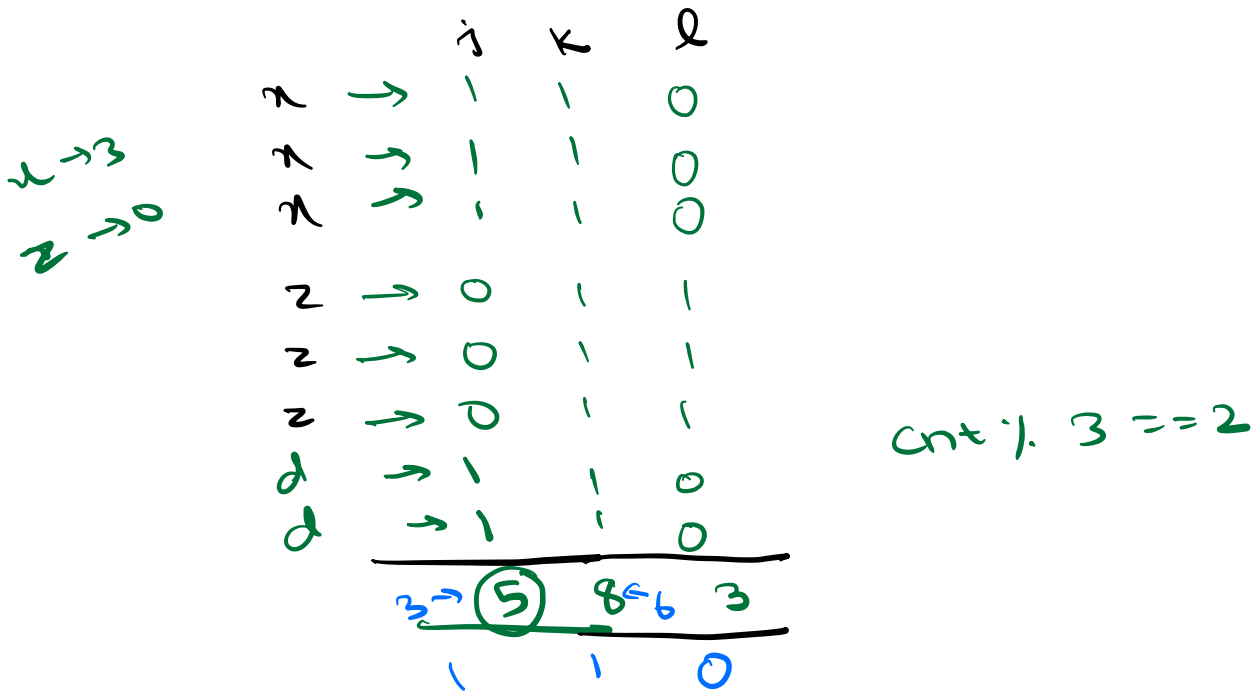
```
int ans = 0

for (i = 0 ; i < 32 ; i++) {
    int cnt = 0
    for (j = 0 ; j < n ; j++) {
        if (ar[j] & (1<<i) != 0)
            cnt++
    }
    if (cnt % 3 == 1)
        ans = ans | (1<<i)
}
return ans
```

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

What if ? every dem comes thrice except unique de comes twice

3, 3, 1, 1, 1, 2, 2, 3 ans = 2



cnt of set bits % 3

what if ! every elem comes 5 times
unique elem comes 3 times

cnt of set bits → groups of 5
(ignoring uniques)

bit pos
x

cnt % 5 == 3 → T → unique elem 1
→ F → 0

10:45

3. Given integer array, all elements occur twice except two. Find these two elements

A[] = 4, 5, 4, 1, 6, 6, 5, 2 ans = 1, 2

$4 \wedge 4 \wedge 5 \wedge 5 \wedge 6 \wedge 6 \wedge 1 \wedge 2$

$0 \wedge 0 \wedge 0 \wedge 1 \wedge 2$

xor → $1 \wedge 2$

2, 5 →
1, 6 → 7
3, 4 →

xor → 7

A[] = 3, 7, 6, 7, 3, 8, 9, 9 ans = 6, 8

ans = 11, 17

10	8	8	9	12	9	6	11	10	6	12	17
1010	1000	1000	1001	1100	1001	0110	1011	1010	0110	1100	10001

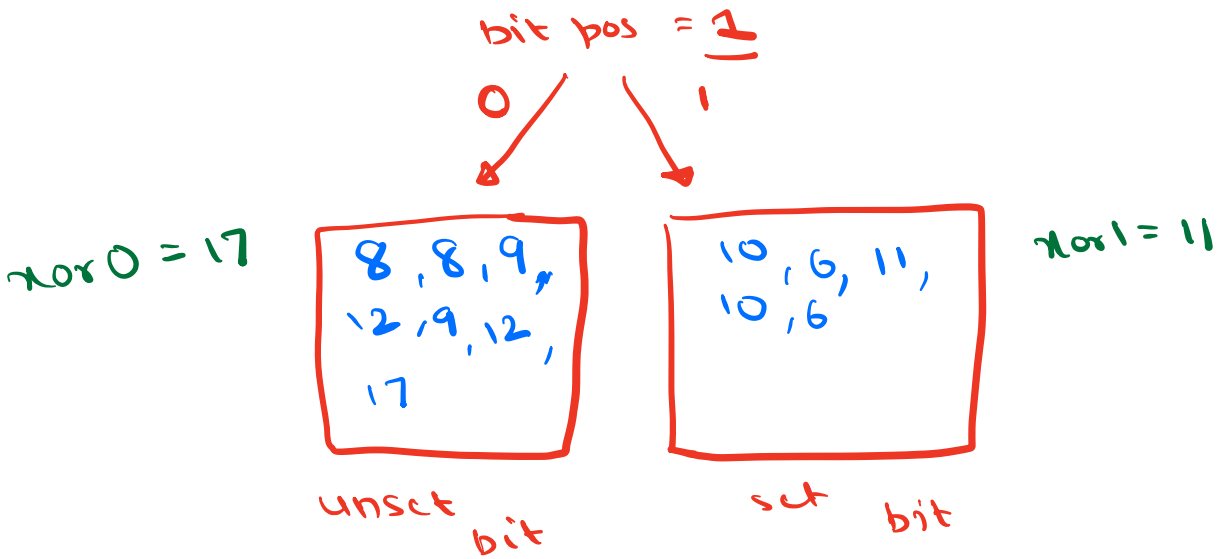
$10 \wedge 8 \wedge 8 \wedge 9 \wedge 12 \wedge 9 \wedge 6 \wedge 11 \wedge 10 \wedge 6 \wedge 12 \wedge 17$

$10 \wedge 10 \wedge 8 \wedge 8 \wedge 9 \wedge 9 \wedge 12 \wedge 12 \wedge 6 \wedge 6 \wedge 11 \wedge 17$

$0 \quad 0 \quad 0 \quad 0$

xor of arr[] → xor of unique elem

	4	3	2	1	0
xor	1	0	1	0	1
17	1	0	0	0	1
	1	1	0	1	0



①
```
int xorall = 0
for (i = 0 ; i < n ; i++) {
    xorall = xorall ^ arr[i]
}
```

②
```
// Look for a set bit in xorall
for (pos = 0 ; pos < 32 ; pos++) {
    if (xorall & 1<<pos != 0)
        break
}
```

// pos → bucket 1 → bucket 0

③ int num1 = 0 , num2 = 0

```
for (i = 0 ; i < n ; i++) {
    if ((arr[i] & (1<<pos)) != 0)
        num1 = num1 ^ arr[i]
    else
        num2 = num2 ^ arr[i]
}
return num1, num2
```

TC : O(N)

SC : O(1)

---

4. Given N array elements, choose 2 indices (i, j) such that i != j and (arr[i] & arr[j]) is maximum

ans (0,4)

max & val = 5

0	1	2	3	4
5	4	6	8	5

0	1	2
27	18	20

16 (0–2), 18 (0–1), 16 (1–2)

18 → pair 27, 18 → ans → (0, 1)

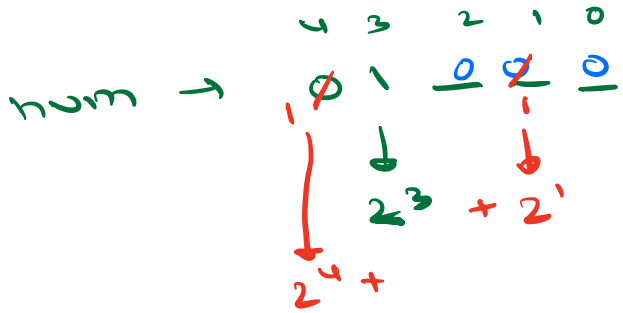
5,4
5,6
5,8
5,5 → 5

4,6
4,8
4,5

6 0 1 1 0
& 8 1 0 0 0
0 0 0 0

6 & 8 → 0

Magnitude → set bit

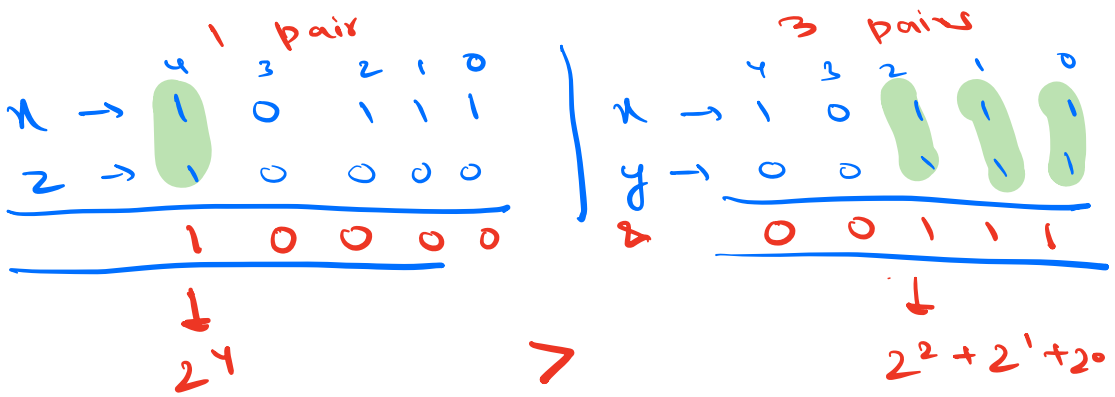


① & value → good no. of set bits

② & value → set bit → A and B have set bit

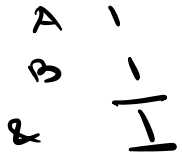
1 & 1 → 1

2 nos. have more and more set bits at common bys

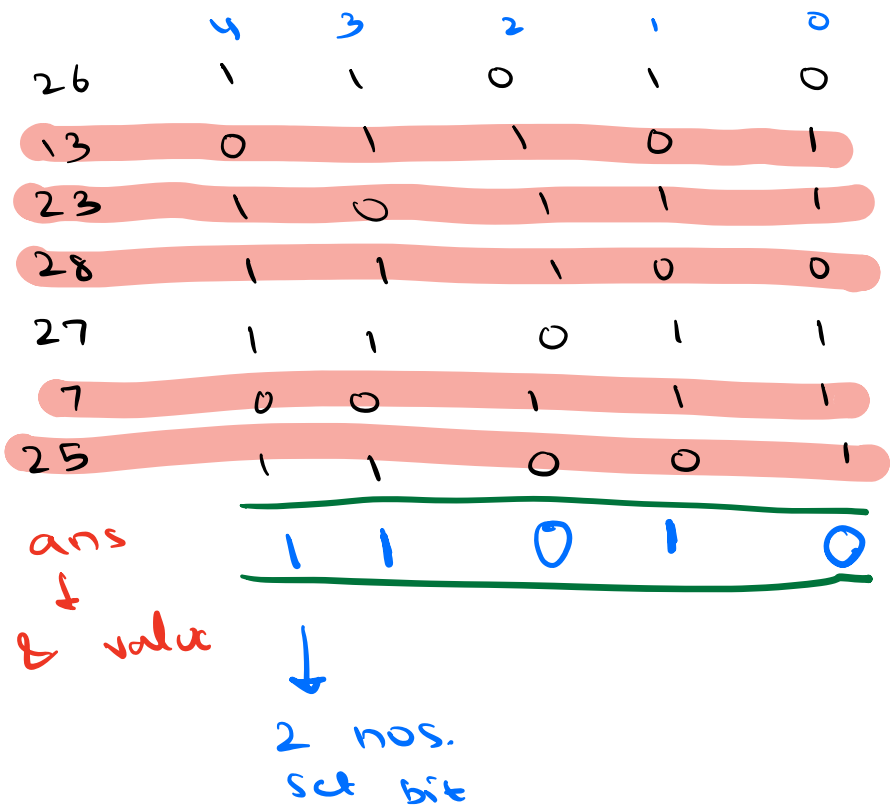


Left side

MSB → LSB



26, 13, 23, 28, 7, 25



mark value → 26

26, 27

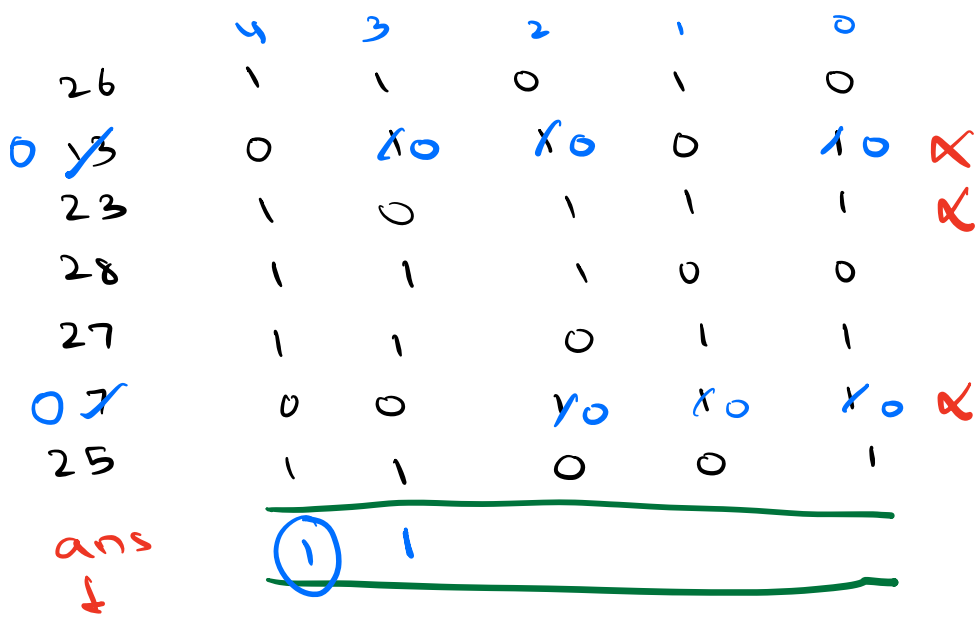
2 nos. set bit

4 3 2 1 0
1 1 0 1 0

$2^4 + 2^3 + 2^1$

16 + 10

26



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

```
int ans=0
for (i=31 ; i≥0 ; i--) {
    int cnt = 0
    for (j=0 ; j<n ; j++) {
        if((ar[j] & (1<<i)) != 0)
            cnt++
    }
    if (cnt ≥ 2) {
        ans = ans | (1<<i)
        for (j=0 ; j<n ; j++) {
            if((ar[j] & (1<<i)) == 0)
                ar[j]=0
        }
    }
}
// ans → max & value

// no. with non 0 value →
   can be pair
```

5. Count all pairs with MAX & value

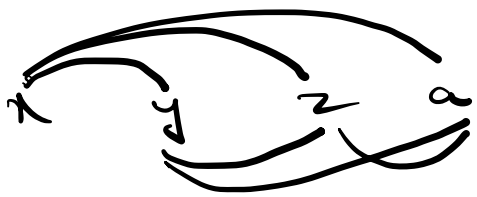
N elements → non zero

pairs $\frac{N(N-1)}{2}$

TC: O(N)

SC: O(1)

4 de



3 2 1

$\frac{3 \times 4}{2}$ ← 3 2 1

= 6 pairs
