Todays Content

1. Sum of xor of all pairs
2.

pow a^n using bit manipulation

18 Given an ar[N] find sum of xor of all pairs

Eg1:

	0	1	2	3
ar[3] = {	4	2	5	3 } return 56

#pairs

(4^4)	(4^2)	(4^5)	(4^3)
(2^4)	(2^2)	(2^5)	(2^3)
(5^4)	(5^2)	(5^5)	(5^3)
(3^4)	(3^2)	(3^5)	(3^3)

#Idea1: Generate all xor pairs & calculate xor for each pair and add in sum;

Given ar[N]; TC: O(N²) SC: O(1)

```
int sum=0;
for(int i=0;i<N;i++){
    for(int j=0;j<N;j++){
        sum = sum + ar[i]^ar[j]
    }
}
return sum;
```

#Idea2: Iterate in lower or upper triangle calculate sum of xor of pairs & return sum*2;

	0	1	2	3
ar[3] = {	4	2	5	3 } return 56

i \ j:	0	1	2	3	Uppr
0	(4^4)	(4^2)	(4^5)	(4^3)	i=0, j=1
1	(2^4)	(2^2)	(2^5)	(2^3)	i=1, j=2
2	(5^4)	(5^2)	(5^5)	(5^3)	i=2, j=3
3	(3^4)	(3^2)	(3^5)	(3^3)	

Given ar[N]; TC: O(N²/2) SC: O(1)

```
int sum=0;
for(int i=0;i<N;i++){
    for(int j=i+1;j<N;j++){
        sum = sum + ar[i]^ar[j]
    }
}
return sum*2;
```

#Idea3: Contribution: Add contribution of each element.
Update: Add contribution of each bit.

	0	1	2	3	4	5
Ex: arr[] = {	15	24	11	19	28	9 }

#Dry Run:

	2^4	2^3	2^2	2^1	2^0
	4	3	2	1	0
15 =	0	1	1	1	1
24 =	1	1	0	0	0
11 =	0	1	0	1	1
19 =	1	0	0	1	1
28 =	1	1	1	0	0
9 =	0	1	0	0	1

i^{th} bit : 1 | i^{th} bit : 0

#Bits:	#Set	#UnSet	#Pairs	#Contribution
0	{15 11 19 9} : 4	{24 30} = 2	4*2*2 = 16	$16*2^0$
1	{15 11 19} : 3	{24 28 9} = 3	3*3*2 = 18	$18*2^1$
2	{15 28} : 2	{24 11 19 9} = 4	2*4*2 = 16	$16*2^2$
3	{15 24 11 28 9} : 5	{19} = 1	5*1*2 = 10	$10*2^3$
4	{24 19 28} : 3	{15 11 9} = 3	3*3*2 = 18	$18*2^4$

#Add contribution of each bit. Final ans = 484

#Idea2:

Iterate on each bit: [0 31] : i

#Calculate contribution of bit position i.

Iterate on arr[] & calculate no. of elements with i^{th} bit set c.

#Set = c, #UnSet = N-c #Pairs = c*(N-c)*2;

#Contribution of i^{th} bit = Pairs * 2^i

}

```
long pairSum [vector<int> &arr] {
    long sum=0;
    int N= arr.size()
    for(int i=0; i<32; i++) { # i = bit position
        # For i: Calculate no: of elements with i^N bit set.
        int c=0;
        for [ int j=0; j<N; j++) { # j: arr[] element
            if [ (arr[j]>>i)&1 == 1) {
                c++;
            }
        }

        # Set = c, # Unset = N-c  # Pairs = 2*c*(N-c)
        # Contribution = 2*c*(N-c)*2^i
        sum = sum + 2*c*(N-c)*(1<<i)
    }

    return sum;
}
```

TODO: Sum of OR of all pairs ✓ Medium

TODO: Sum of AND of all pairs ✓ Simple

28 Given a, n calculate & return a^n

Eg: a n

3 5

2 4

#Ideal:

```
long power(int a, int n){
|
3
```

28 Given ar[N] it contains all elements from 1..N.

1 element from 1 to N repeats

1 element from 1 to N missing

Return both repeat & missing element

Note: No Extra space, No modifying array.

Constraints:

$1 <= N <= 10^6$

$1 <= ar[i] <= N$.

Ex:

	Missing	repeat
ar[5] = { 2 2 1 4 5 }	3	2
ar[7] = { 1 3 6 5 4 6 7 }	2	6

Idea: With ar[] elements consider all elements from 1..N

If we consider entire data:

Every ar[i] element repeats twice except 2 elements

Those 2 elements are repeating & missing elements: TODO

	Given arr[]	Expected Data
ar[7] =	{ ✗ ✗ 6 ✗ ✗ ✗ ✗ }	{ ✗ 2 ✗ ✗ ✗ ✗ ✗ }