Content

- Bitwise operators & properties.
- Check-Bit
- Single Element - I
- Single Element - II
- Single Element - III
- Max AND Pair.

Bitwise Operators

sam sam puppy shame

a	b	a|b	a&b	a^b	~a
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

a = 29 : 0 0 0 1 1 1 0 1

b = 18 : 0 0 0 1 0 0 1 0

a & b : 0 0 0 1 0 0 0 0 → [16]

a | b : 0 0 0 1 1 1 1 1 → [31]

a ^ b : 0 0 0 0 1 1 1 1 → [15]

Properties

$a \& b = b \& a$

$a | b = b | a$

$a \wedge b = b \wedge a$

// Commutative prop.

$a \& b \& c = (b \& a) \& c = (c \& a) \& b$

$a | b | c = (a | b) | c = a | (b | c)$

$a \wedge b \wedge c = (a \wedge b) \wedge c = a \wedge (b \wedge c)$

// Associative Property

$a = 10$

$$\begin{array}{cccc} 1 & 0 & 1 & 0 \\ & & \& & 1 \\ \hline 0 & 0 & 0 & 0 \end{array}$$

$a = 11$

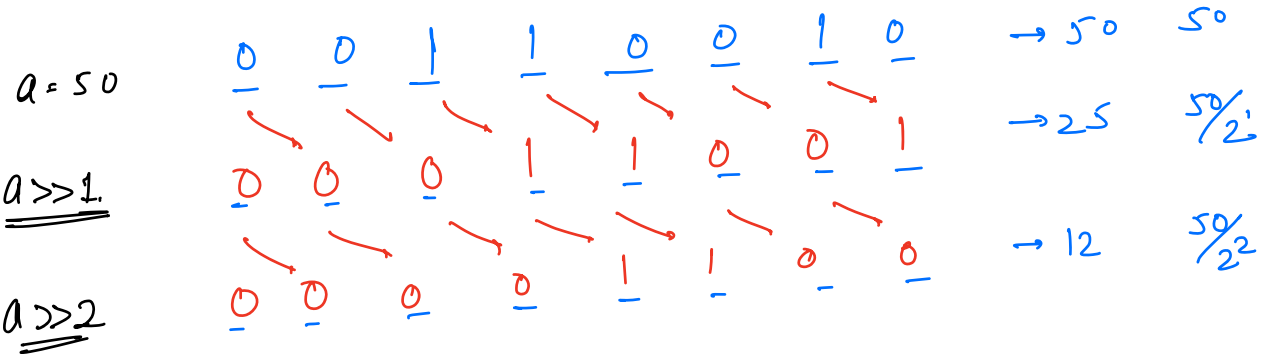
$$\begin{array}{cccc} 1 & 0 & 1 & 1 \\ & & \& & 1 \\ \hline 0 & 0 & 0 & 1 \end{array}$$

$a \& 1$

- $= 0$ → 0^{th} bit of a is 0 ⇒ Even number
- $= 1$ → 0^{th} bit of a is 1 ⇒ odd number.

$a \& 0 = 0$	$a \| 0 = a$	$a \wedge 0 = a$
$a \& a = a$	$a \| a = a$	$a \wedge a = 0$

Left & Right Shift →



$a >> N. = a/2^N$

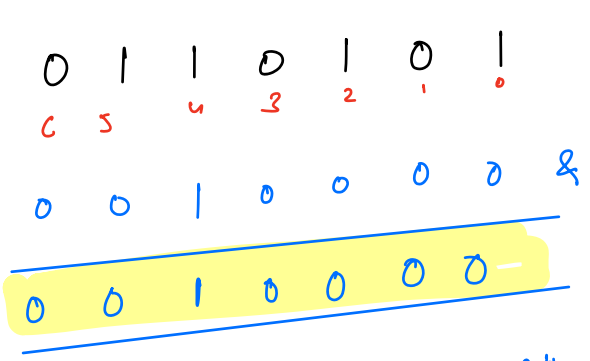
$a << N. = a * 2^N$ [overflow]

$1 << N = 2^N$

Q: Given a number N & i. Check if ith-bit is set.

set → 1 unset → 0

① N = 53



i=2 → set

i=3 → unset.

⇒ 4th bit is set

code:→

N=53 , i=5

```
boolean checkBit(N, i){
    if(((1<<i) & N) == 0){
        //ith bit is unset
        return false;
    }
    else{
        return true;
    }
}
```

53 → 1 1 0 1 0 1

(1<<5) → 1 0 0 0 0 0 &

1 0 0 0 0 0

[T.C → O(1)
S.C → O(1)]

[# → try doing this with right shift operator.]

Q: Given N elements, every element repeats twice except one. Find unique element.

arr[7]: { 3 2 3 7 2 8 7 } ans = 8.

// ans = Xor of all the elements.

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

T.C → O(N)
S.C → O(1)

Q) Every element repeats thrice except one. Find unique element.

Constraints : $1 <= N <= 10^6$, $1 <= arr[i] <= 10^9$

arr[13]: { 5 7 5 4 7 11 11 9 11 7 5 4 4 }

idea-1 for every element, iterate on array and find the frequency of that element.

T.C → $O(N^2)$, S.C → $O(1)$

idea-2 Insert all elements in hashmap & store their frequency. for any elements if freq = 1., that element will be the ans.

T.C → $O(N)$, S.C → $O(N)$

Extra space is not allowed.

idea → take XOR.

⇒ 5^7^5^4^7^11^11^9^11^7^5^4^4 =

= 5^5^5^4^4^4^7^7^7^11^11^11^9

⇒ 5^4^7^11^9 = 4.

0 1 0 1	(5)
0 1 0 0	(4)
0 0 0 1	
0 1 1 1	(7)
0 1 1 0	(11)
1 0 1 1	
1 1 0 1	
1 0 0 1	(9)
0 1 0 0	

arr[13]: { 5 7 5 4 7 11 11 9 11 7 5 4 4 }
0 1 2 3 4 5 6 7 8 9 10 11 12

	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
	4	9	6	10
	(4%3)	(9%3)	(6%3)	(10%3)
ans →	1	0	0	1

idea :-
[Count set-bits at every bit position.]

	3	2	1	0
11 →	1	0	1	1
11 →	1	0	1	1
5 →	0	1	0	1
11 →	1	0	1	1
ans :-	0	1	0	1

pseudo-code.

```
ans = 0   // Initially all bits are '0' in ans.

for( i = 0; i < 31 ; i++){
    //find count of set-bits at ith- bit position.
    count = 0
    for( j = 0; j < N ; j++){
        if( checkBit( arr[j], i) == true){
            count ++ ;
        }
    }
    if( count % 3 == 1){
        //set the ith bit in the ans.
        ans = (ans | (1<<i)) ;
    }
}

return ans;
```

(revise it)

T.C → O(N)
S.C → O(1)

[Break → 10:37 → 10:42]

Q) Given N elements every element repeats twice except 2 unique elements. find two unique elements.

Eg: arr[6] → { 3 6 4 4 3 8 } ans = 6, 8

arr[4] → { 4 9 9 8 } ans = 4, 8

idea → ~~3^6^4^4^3^8~~ = 6^8 = 14.

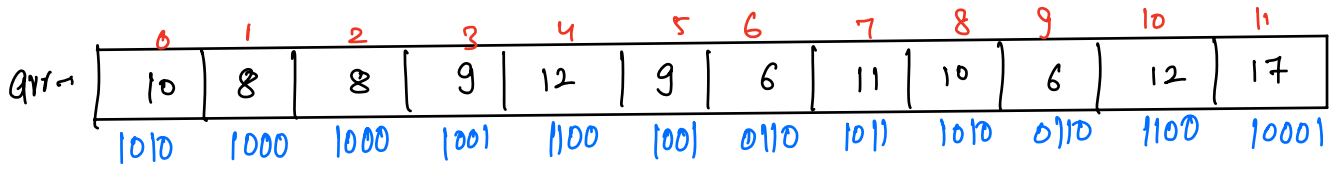
⇓

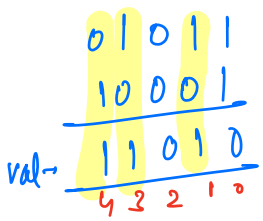
[How to extract unique no's from this val]

Eg → [5 2 7 2 3 3]

val = 5^7 = 2

```
0 1 1 1
0 1 0 1
-------
0 0 1 0
-------
```





obs.→ bits are different at [1, 3, 4] for 2 unique elements.

Split the entire array on the basis of 1st bit

set

unset.

[10, 6, 11, 10, 6,]

S1

[8, 8, 9, 12, 9, 12, 17]

S2

Xor of S1 → 11

Xor of S2 → 17

pseudo-code val = 0

Step-1 Take XOR of all the elements

```
for( i = 0; i < N; i++){
    val = (val ^ arr[i]
}
```

Step-2 Find set-bit position in the val

```
pos = 0
for( i = 0; i < 31; i++){
    if ( checkBit( val, i) == true){
        pos = i;
        break;
    }
}
```

Step-3 Split the array on the basis of pos-th bit

```
s1 = 0,  s2 = 0
for( i = 0; i < N; i++){
    if ( checkBit( arr[i], pos) == 1){
        s1 = (s1 ^ arr[i]);
    }
    else{
        s2 = (s2 ^ arr[i]);
    }
}
print(s1, s2);
```

T.C → O(N)
S.C → O(1)

Maximum AND Pair

Given N +ve array elements. Find maximum value of (arr[i] & arr[j]) where $i \neq j$.

arr[3]: [27 18 20]

(27 & 18) → 18

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

(27 & 20) → 16

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

(18 & 20) → 16

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

arr[4] → [21 18 24 17]
0 1 2 3

ans = 17.

```
21 → 1 0 1 0 1
18 → 1 0 0 1 0
     ---------
     1 0 0 0 0
```

```
21 → 1 0 1 0 1
24 → 1 1 0 0 0
     ---------
     1 0 0 0 0
```

```
21 → 1 0 1 0 1
17 → 1 0 0 0 1
     ---------
     1 0 0 0 1
```

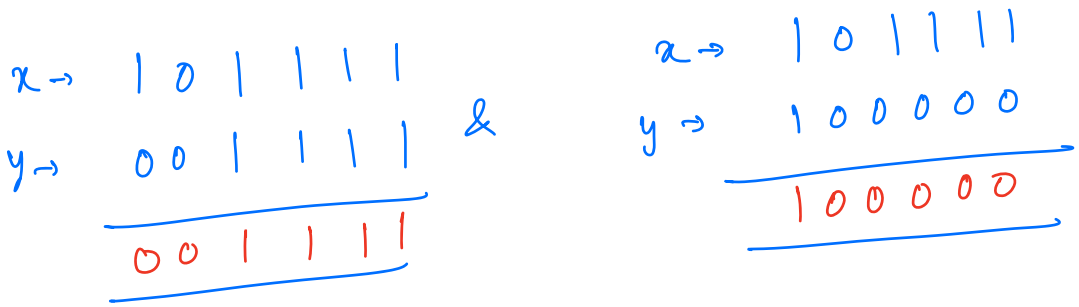
```
18 → 1 0 0 1 0
24 → 1 1 0 0 0
     ---------
     1 0 0 0 0
```

```
18 → 1 0 0 1 0
17 → 1 0 0 0 1
     ---------
     1 0 0 0 0
```

```
24 → 1 1 0 0 0
17 → 1 0 0 0 1
     ---------
     1 0 0 0 0
```

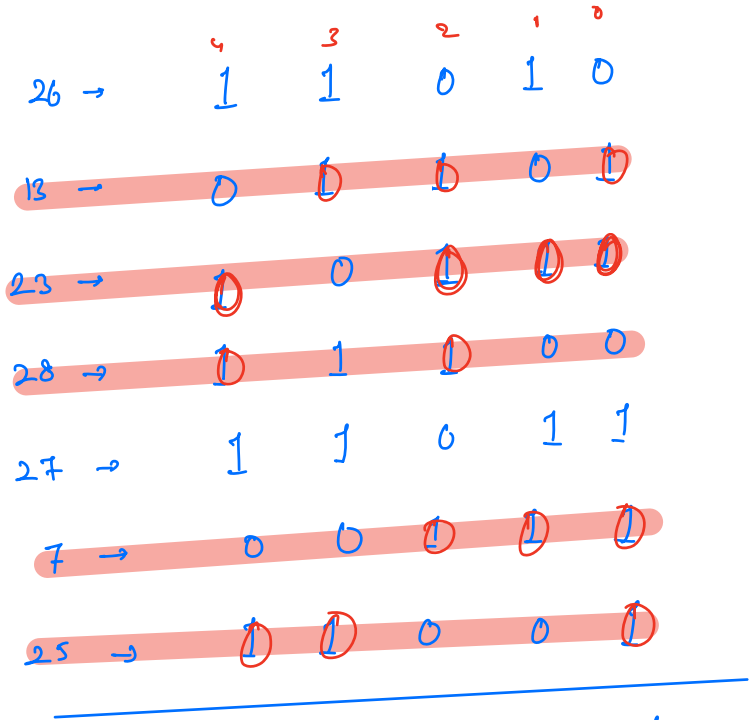
idea-1 Consider all pairs where $i \neq j$

T.C → $O(N^2)$, S.C → $O(1)$



(ccc) → bit-mask

arr[6] : [26 13 23 28 27 7 25]



count of set-bits	5	4	1	2	1
			x par		x par

ans :- [1 1 0 1 0]

$2^4 + 2^3 + 2^1 = 26$

#code→

```
ans = 0
for( i = 30; i ≥ 0; i-- ){
    //count of set bits at ith bit position
    count = 0
    for( j = 0; j < N; j++){
        if( checkBit( arr[j], i) == true){
            count ++;
        }
    }
    if ( count ≥ 2) {
        //set ith-bit in the ans
        ans = (ans | (1<<i));
        // make all element with ith bit unset → 0
        for( j = 0; j < N; j++){
            if( checkBit( arr[j], i) == false){
                arr[j] = 0;
            }
        }
    }
}
return ans;
```

T.C → O(N)
S.C → O(1)

- bit by bit [Hint]
- optimised T.C & S.C ⇒ think of bit manipulation

①
```
for( i=0; i<N; i+=K){
    for( j=0; j<K; j++){
    }
}
```
→ O(N)

②
```
i = 0, j = 0

while( i < N){
    j++;
    if( j == N){
        j = 0
        i++;
    }
}
```