Agenda

1) Revision quizzes

2) unique element (rest all coming thrice)

3) every element will occur twice except two ele

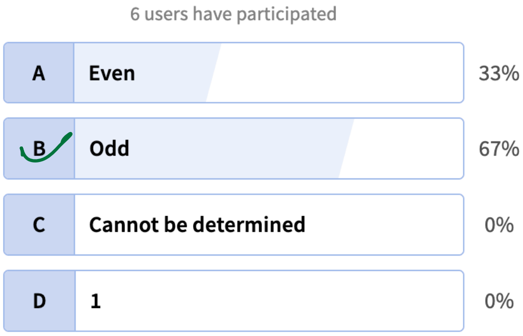
} very important

2.5 hrs

class notes revision → 25 mins step

solving assign

If the last bit(0th bit) of a number is 1 (rest can be anything), then the number is?



6 users have participated

A	Even	33%
[✓ B]	Odd	67%
C	Cannot be determined	0%
D	1	0%

25 → 1100$\underline{\underline{1}}$

24 → 1100$\underline{\underline{0}}$

What is the time complexity of the operation to check whether the i-th bit of a number with N bits is set or not?

no. $\underline{\underline{A}}$

→ check ith bit : A & (1<<i) — 0 (off) / non-zero (on)

→ set ith bit : A | (1<<i)

→ toggle ith bit : A^(1<<i)

What is the time complexity to count the total number of set bits of an integer with $\underline{\underline{N}}$ bits ?

TC: O (no. of bits)

↓

O(N)

(Based on last class ques logic)

↓

no. bits = 32

32 itr ⇒ O(1)

Q. Given A[], all elements occurs thrice except one element. Find the unique element.

A = [4 5 5 4 1 6 6 4 5 6]   ans = 1
0 1 2 3 4 5 6 7 8 9

A = [5 7 5 9 7 11 11 7 5 11]   ans = 9
0 1 2 3 4 5 6 7 8 9

→ Brute-force : two loops

for (go to every element)

for (find freq of ele by travelling complete)

TC: $O(n^2)$

Demand : TC → $O(n)$

SC → $O(1)$

$$A = \begin{array}{cccccccccc} [5 & 7 & 5 & 9 & 7 & 11 & 11 & 7 & 5 & 11] \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

$$\text{ans} = \overset{1}{\cancel{0}}\,0\,0\,\overset{1}{\cancel{0}}$$
$$= \underline{9}$$

	3	2	1	0
5 →	0	1	0	1
7 →	0	1	1	1
5 →	0	1	0	1
9 →	1	0	0	1
7 →	0	1	1	1
11 →	1	0	1	1
11 →	1	0	1	1
7 →	0	1	1	1
5 →	0	1	0	1
11 →	1	0	1	1
	4	6	6	10

1 0 0 1

→ 9

0 to 31

0th bit => 10 (10%3 == 1)

1th bit => 6 (6%3 == 0)

2nd bit → 6 (6%3 == 0)

3rd bit → 4 (4%3 == 1)

```
                         N
function  uniqueNum ( A[ ] )

    ans = 0;

    for ( i → 0 to 31 ) {

        count = 0;
        // in how many elements of A[ ], ith bit is "on"

        for ( j → 0 to N-1 ) {

            if ( ( A[j] & (1<<i) ) != 0 ) {
                count++;
            }
        }

        if ( count % 3 == 1 ) {
            // set ith bit in ans

            ans = ans | (1<<i);
        }
    }

    return ans;
}
```

TC : O(N)

SC : O(1)

```
function uniqueNum ( A[] )  ↷N
    ans = 0;

    for ( i -> 0 to 31) {
        count = 0;
        // in how many elements of A[], ith bit is "on"
        for (j -> 0 to N-1) {
            if ( (A[j] & (1<<i)) != 0) {
                count++;
            }
        }
        if ( count % 3 == 1) {
            // set ith bit in ans
            ans = ans | (1<<i);
        }
    }
    return ans;
}
```

A = [8 5 8 8 6 5 5]

A =	8	5	8	8	6	5	5
	0	1	2	3	4	5	6

		3	2	1	0
8	→	1	0	0	0
5	→	0	1	0	1
8	→	1	0	0	0
8	→	1	0	0	0
6	→	0	1	1	0
5	→	0	1	0	1
5	→	0	1	0	1
count :		3	4	1	3

ans = $0\,\overset{1}{\cancel{0}}\,\overset{1}{\cancel{0}}\,0$

= 6

to do : dry run this eg. using code

A =	5	7	5	9	7	11	11	7	5	11
	0	1	2	3	4	5	6	7	8	9

Q. Given an A[] in which all elements will occur twice except two elements. Find those two elements.

$$A = \begin{array}{cccccccc} [4 & 5 & 4 & 1 & 6 & 6 & 5 & 3] \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \implies 1 \quad 3$$

$$A = \begin{array}{cccccccc} [9 & 5 & 6 & 7 & 6 & 2 & 7 & 5] \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \implies 2 \quad 9$$

Brute force : $O(n^2)$ {two loops}

↓ better

$O(n\log n)$ {sorting}

$$A = \begin{array}{cccccccc} [9 & 5 & 6 & 7 & 6 & 2 & 7 & 5] \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

$$\text{xorAll} = 9 \wedge \cancel{5} \wedge \cancel{6} \wedge \cancel{7} \wedge \cancel{6} \wedge 2 \wedge \cancel{7} \wedge \cancel{5}$$

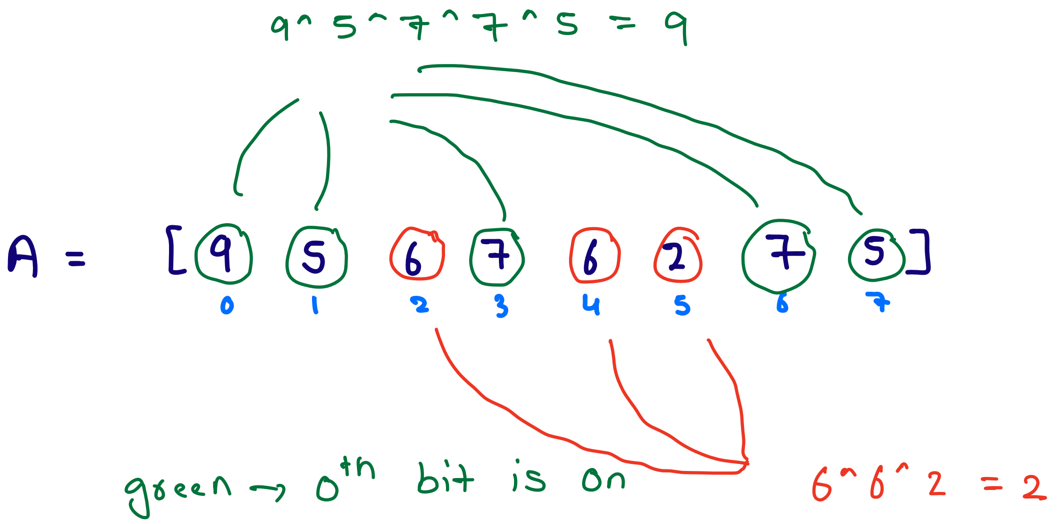
$$= 9 \wedge 2$$

$$= 11$$

xor of two numbers, find those numbers.

$$\begin{array}{rcccc|c|l} & 1 & 0 & 0 & 1 & (9) \\ \text{xor} & 0 & 0 & 1 & 0 & (2) \\ \hline & 1 & 0 & 1 & 1 & (11) \\ & 3 & 2 & 1 & 0 & \end{array}$$

0th bit in one of the no. is on and in other no. it off.



green → 0th bit is on

red → 0th bit is off

→ A = [9 5 6 7 6 11 7 5]

0 1 2 3 4 5 6 7

i) $\boxed{\text{xorAll}}$ = xor of entire array = $9 \wedge \cancel{5} \wedge \cancel{6} \wedge \cancel{7} \wedge \cancel{6} \wedge 11 \wedge \cancel{7} \wedge \cancel{5}$

$= 9 \wedge 11$

$= 2$

```
     1001
xor  1011
     ----
     00[1]0
     32 1 0
```

ii) In variable "xorAll" find the first bit which is on

0 to 31 ⇒ ans : 1

iii) categorising the array in two parts

ele in which $\boxed{1^{st}}$ bit is on

ele in which $\boxed{1^{st}}$ bit is off

A =	[9	5	6	7	6	11	7	5]
	0	1	2	3	4	5	6	7

. . . .	__	__	__	__
	3	2	1	0

num1
(1st bit is on) = 6 ^ 7 ^ 6 ^ 11 ^ 7 = 11

num2
(1st bit is off) = 9 ^ 5 ^ 5 = 9

```
                      N
function  solve ( A[] ) {

   //find xor of all elements

   xorAll = 0;

   for (i -> 0 to N-1) {
   |    xorAll = xorAll ^ A[i];
   }

   // in xorAll, find the posn of rightmost set bit

   pos = -1;

   for (i -> 0 to 31) {
   |    if ( ( xorAll & (1<<i)) != 0) {
   |         pos = i;
   |         break;
   |    }
   }

   // categorise array into 2 parts based of bit at pos

   num1 = 0;  // contribution of ele in which bit at pos is on

   num2 = 0;  // contribution of ele in which bit at pos is off

   for (i -> 0 to N-1) {
   |    if ( (A[i] & (1 << pos)) != 0) {
   |         num1 = num1 ^ A[i];
   |    }
   |    else {
   |         num2 = num2 ^ A[i];
   }    }

   print (num1);
   print (num2);
}
```

TC : O(N)

SC : O(1)

```
           N
function solve (A[]) {
    //find xor of all elements
    xorAll = 0;
    for (i -> 0 to N-1) {
        xorAll = xorAll ^ A[i];
    }
    // in xorAll, find the posⁿ of rightmost set bit
    pos = -1;
    for (i -> 0 to 31) {
        if ((xorAll & (1<<i)) != 0) {
            pos = i;
            break;
        }
    }
    // categorise array into 2 parts based of bit at pos
    num1 = 0; // contribution of ele in which bit at pos is on
    num2 = 0; // contribution of ele in which bit at pos is off
    for (i -> 0 to N-1) {
        if ((A[i] & (1<<pos)) != 0) {
            num1 = num1^A[i];
        }
        else {
            num2 = num2^A[i];
        }
    }
    print(num1);
    print(num2);
}
```

$$A = \begin{array}{cccccccc} [\,5 & 9 & 5 & 6 & 6 & 2 & 13 & 2\,] \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

1) $xorAll = \cancel{5}\,\hat{}\,9\,\hat{}\,\cancel{5}\,\hat{}\,\cancel{6}\,\hat{}\,\cancel{6}\,\hat{}\,\cancel{2}\,\hat{}\,13\,\hat{}\,\cancel{2}$

$= 9\,\hat{}\,13 = 4$

$$\begin{array}{lcccc} 9 \rightarrow & 1 & 0 & 0 & 1 \\ 13 \rightarrow & 1 & 1 & 0 & 1 \\ \hline 4 \rightarrow & 0 & \boxed{1} & 0 & 0 \\ & 3 & \boxed{2} & 1 & 0 \end{array}$$

2) $pos = 2$

$$3)\ A = \begin{array}{cccccccc} [\,5 & 9 & 5 & 6 & 6 & 2 & 13 & 2\,] \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

$num1 = \cancel{5}\,\hat{}\,\cancel{5}\,\hat{}\,\cancel{6}\,\hat{}\,\cancel{6}\,\hat{}\,13 = 13$

$num2 = 9\,\hat{}\,2\,\hat{}\,2 = 9$

$\underset{3}{\_}\ \underset{\boxed{2}}{\_}\ \underset{1}{\_}\ \underset{0}{\_}$

to do: dry run on one eg.