Today's agenda

↳ left shift + 4 Problems

↳ Single element 1

↳ Single element 2

↳ you didn't come this far only to come this far.

* $(1 << 2) \Rightarrow 1 * 2^2 = 4$

$(1 << 4) \Rightarrow 1 * 2^4 = 16$

$(a << N) \Rightarrow a * 2^N$

* **Power of left shift**

① Check if ith bit of given number N is $\overset{1}{\text{set}}$ or $\overset{0}{\text{not}}$!

ex: N = 45 ,

i:2

$N = 45 :$ $\overset{5}{1}\ \overset{4}{0}\ \overset{3}{1}\ \overset{2}{1}\ \overset{1}{0}\ \overset{0}{1}$

i:4

$N = 45 :$ $\overset{5}{1}\ \overset{4}{0}\ \overset{3}{1}\ \overset{2}{1}\ \overset{1}{0}\ \overset{0}{1}$

//idea1

↳ Simply Convert the decimal value into binary and Check the Corresponding bit index.

//idea2

i:2

$N = 45 :$ $\overset{5}{1}\ \overset{4}{0}\ \overset{3}{1}\ \overset{2}{1}\ \overset{1}{0}\ \overset{0}{1}$

&

$x = 4 :$ $0\ 0\ 0\ 1\ 0\ 0$

$0\ 0\ 0\ 1\ 0\ 0$ $!= 0$

i:4

$N = 45 :$ $\overset{5}{1}\ \overset{4}{0}\ \overset{3}{1}\ \overset{2}{1}\ \overset{1}{0}\ \overset{0}{1}$

&

$x = 2^4 = 16 :$ $0\ 1\ 0\ 0\ 0\ 0$

$0\ 0\ 0\ 0\ 0\ 0$ $= 0$

N & n → Non . Zero ⇒ Set (1)

N & n → zero ⇒ unset (0)

↓

(1<<i)

i:2

↓

1<<2 ⇒ 2^2:4 : 100

i:4

→ 1<<4 = 2^4=16

↓

10000

//Psuedo Code

bitwise operator

```
boolean checkbit(int N, int i){
    if (N & (1<<i) ==0){
        return false        //false == unset
    } else {
        return true         //true = set.
    }
}
```

T.C: O(1)

S.C: O(1)

(II) Set the ith bit

i=2

	5	4	3	2	1	0
N = 45 :	1	0	1	1	0	1
or (1<<2)	0	0	0	1	0	0
	1	0	1	1	0	1

1 dominated

i=4

	5	4	3	2	1	0
N = 45 :	1	0	1	1	0	1
or (1<<4)	0	1	0	0	0	0
	1	1	1	1	0	1

```
int setbit(int N, int i){
    int ans = N | (1<<i);
    return ans;
}
```

N:45 i:4

1<<4 = 16

N:45= 101101

16= 010000

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

⇓

decimal= $2^5+2^4+2^3+2^2+2^0$

= ✓

(III) flip ith bit (toggle ith bit)

i=2

```
            5 4 3 2 1 0
N = 45 :    1 0 1 1 0 1
  xor
  1<<2      0 0 0 1 0 0
            -----------
            1 0 1 0 0 1
```

Same Same Puppy Shame

i=4

```
            5 4 3 2 1 0
N = 45 :    1 0 1 1 0 1
  xor
  1<<4      0 1 0 0 0 0
            -----------
            1 1 1 1 0 1
```

```
int flipbit(int n, int i){
    int ans = n^(1<<i);
    return ans;
}
```

(IV) unset ith bit of a number.

i=2

```
            5 4 3 2 1 0
N = 45 :    1 0 1 1 0 1
            -----------
            1 0 1 0 0 1
```

i=4

```
            5 4 3 2 1 0
N = 45 :    1 0 1 1 0 1
            -----------
            1 0 1 1 0 1
```

T.C: O(1)

S.C: O(1)

```
if ( checkbit(n,i) == true){
    int ans = n^(1<<i);
    return ans;
}
else {
    //do nothing
}
```

Q) Count number of set bits in N.

int N → Atmost 32 bits

```
     31                                                  0
N:   _   _   _   _   _   .  .  .  .  .  .  _  _  _   _
```

```
                                                 i
                                                 ↙
          31 . . . . . . . . . . .  5   4   3   2   1   0
N = 45 :  0   0   0   0   0   0     1   0   1   1   0   1
```

```
int countsetbits (int N) {
    int ans=0;
    for (int i=0; i<32; i++){
        if (checkBit(N,i) == true) {
            ans ++;
        }
    }
    return ans;
}
```

T.C: O(32) = O(logN)

S.C: O(1)

```
boolean checkbit (int N, int i){
    if (N & (1<<i) ==0){
        return false     //false == unset
    } else {
        return true      //true = set.
    }
}
```

Tracing

```
int countSetBits(int N){
    int ans=0;
    for(int i=0; i<32; i++){
        if(checkBit(N,i)==true){
            ans++;
        }
    }
    return ans;
}
```

ans: ~~0~~1

```
             31 . . . . . . . . .  5  4  3  2  1  0
                                            ↙i
N = 45 :  0   0  0  0  0  0  1  0  1  1  0  1
```

```
boolean checkBit(int N, int i){
    if((N & (1<<i))==0){
        return false  //false == unset
    } else {
        return true  //true = set.
    }
}
```

```
N:45 :        0 0  0 1  0  1  1  0  1
i = 1 : 2^1 = 2 :   0 0  0  0  0  1  0
              ___________________________
                0 0  0  0  0  0  0
```

Q) Single element 1 → {google, microsoft, Amazon}

↳ Every element repeats twice except 1, find the unique element.

Ex1: arr[7]: {4 2 4 9 2 8 9} → ans: 8

//idea1

↳ use Hashmap and Count occ. of each element.

T.C: O(N) S.C: O(N)

//idea2

↳ $A \wedge 0 = A$ $A \wedge A = 0$

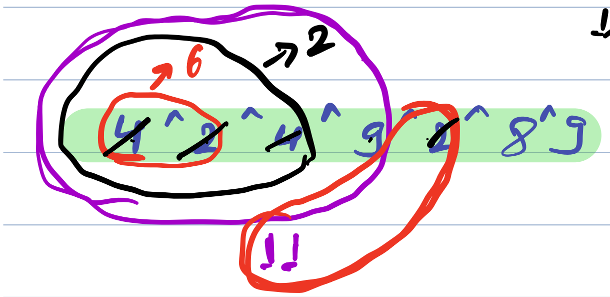
$A \wedge B = B \wedge A$ ✓

arr[7]: {4 2 4 9 2 8 9}

⇓

$\overset{0}{\cancel{2 \wedge 2}} \wedge \overset{0}{\cancel{4 \wedge 4}} \wedge \overset{0}{\cancel{9 \wedge 9}} \wedge 8 = 8$

⇓



$= 2 \wedge 2 \wedge 4 \wedge 4 \wedge 9 \wedge 9 \wedge 8$

↳ Take xor of all the elements.

//Psuedo code

T.C: O(N)

S.C: O(1)

```
int Singleelement1 (int arr[N]){
        int ans = arr[0];
        for (int i=1; i<N; i++){
            ans = ans ^ arr[i];
        }
        return ans;
}
```

Single element 2

↳ Given arr[N], every element repeats thrice except 1, which comes 1 time find the unique element?

Ex:

```
         0 1 2 3 4 5 6
arr[7]: {7 6 7 5 6 7 6} → 5
```

```
          0 1 2 2 4  5  6 7  8 9
arr[13]: {5 7 5 7 11 11 9 11 7 5} → 9
```

//ideas

↳ use hashmap (Count occ. of each number)

T.C : O(N)   S.C: O(N)

// Taking xor of all element

```
          0 1 2 2 4  5  6 7  8 9
```

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

↳ 9 ^ 11 ^ 7 ^ 5 : 4

//idea 2

↳ Counting the bits

$\begin{array}{cccccccccccccc} & 0 & 1 & 2 & 3 & 2 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ arr[13] : & \{5 & 7 & 5 & 4 & 7 & 11 & 11 & 9 & 11 & 7 & 5 & 4 & 4\} \end{array}$ → 9

i

	5	4	2	2	1	0
5	0	0	0	1	0	1
7	0	0	0	1	1	1
5	0	0	0	1	0	1
4	0	0	0	1	0	0
7	0	0	0	1	1	1
11	0	0	1	0	1	1
11	0	0	1	0	1	1
9	0	0	1	0	0	1
11	0	0	1	0	1	1
7	0	0	0	1	1	1
5	0	0	0	1	0	1
4	0	0	0	1	0	0
4	0	0	0	1	0	0
Count	0	0	4	9	6	10

↓ 4%3=1 ↓

9%3==0

↓ 6%3 =0

↓ 10%3 = 1

3 contrib ⇒ 7

3 contrib ⇒ 11

$\begin{array}{ccccccc} & & & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$

⇓

$2^3 + 2^0 = 9$

//Psuedo code

```
int Singleelement2 (int arr[N]){
        int ans =0;

    for(int i=0; i<32; i++){
            int count=0;
        for(int j=0; j<N; j++){
            if(checkBit (arr[j], i) == true){
                Count ++;
            }
        }

        if (count % 3 == 1 ){
            ans = ans + (1<<i);   → 2^i
        }
        else {
            //do nothing
        }
    }
}
```

T.C: $O(32*N) \simeq O(N)$

S.C: $O(1)$

Tracing

		5	4	3	2	1	0
0	5	0	0	0	1	0	1
1	7	0	0	0	1	1	1
2	5	0	0	0	1	0	1
3	4	0	0	0	1	0	0
4	7	0	0	0	1	1	1
5	11	0	0	1	0	1	1
6	11	0	0	1	0	1	1
7	9	0	0	1	0	0	1
8	11	0	0	1	0	1	1
9	7	0	0	0	1	1	1
10	5	0	0	0	1	0	1
11	4	0	0	0	1	0	0
12	4	0	0	0	1	0	0
		0	0	4		6	10

i

Counts

```
int singleelement2 (int arr[N]){
    int ans =0;
    for (int i=0; i<32; i++) {
        int count=0;
        for (int j=0; j<N; j++) {
            if (checkBit (arr[j], i) == true){
                count ++;
            }
        }
        if (count % 3 == 1){
            ans = ans + (1<<i);
        }
        else {
            //do nothing
        }
    }
}
```

2^i

7, 1

$ans = 0 + 2^0 + 2^3 = 9$

//Extensions

↳ (i) Every element is repeating thrice, but 1 element is repeating 2 times.

Count the bits

3*N → ↓ ans → bit is unset

3*N+2 → ↓ ans → bit is set

(ii) Every element is repeating 4 times except 1.
↳ occ : 1 time.

↳ Count the bits at each index

↳ Count %4 == 1 T.C: O(32*N) ≈ O(N)

↳ xor of all the elements T.C: O(N)

(iii) Every element is repeating 4 times except 1.
↳ occ : 2 times

↳ xor won't work

↳ Count bits → Count % 4 == 2 → bit is set.