Bit Manipulation

Bit operation ✓

Q print integer appearing once

a[] = { 2, 1, 2, 5, 6, 5, 7, 7, 6 }

$a \wedge a = 0$

$0 \wedge a = a$

```
xOR = 0;
for(i=0; i<n; i++){
    xOR = xOR ^ arr[i];
}
cout<<xOR;
```

Q2. swap number using XOR

$a = 5, b = 7$

$a = a \wedge b \quad (5 \wedge 7)$

$b = a \wedge b \quad (5 \wedge \cancel{7} \wedge \cancel{7}) = 5$

$a = a \wedge b \quad (\cancel{5} \wedge \cancel{5} \wedge 7) = 7$

Q3 Given N, print the XOR of all no.s between (1 - N)

i/p → N = 5

ans → $1 \wedge 2 \wedge 3 \wedge 4 \wedge 5 = 1$ (ans)

O(1)

observation:

n	XOR (1^2^...^N)
1	1
2	3
3	0
4	4
5	1
6	7
7	0
8	8

```
if(n%4 == 0){
    ans = n;
}
if(n%4 == 3)
{
    ans = 0
}
if( n%4 == 1)
    ans = 1
if(n%4 == 2)
    ans = n+1;
```

Q Given range (L-R) print XOR (L ^ L+1 ^ L+2 --- R-1 ^ R)

eg $L = 2$
$R = 4$

ans = $2 \wedge 3 \wedge 4$

$= (\cancel{1} \wedge 2 \wedge 3 \wedge 4) \wedge \cancel{1}$

previously computed

TC : O(1)

ans = $L \wedge (L+1) \wedge (L+2) \wedge --- (R-1) \wedge R$

$= (1 \wedge 2 \wedge 3 \wedge -- --- \wedge R) \wedge (1 \wedge 2 \wedge 3 - \wedge (L-1))$

$=$ XOR(1 → R) ^ XOR(1 → L-1)

<u>Use case of '&'</u>

<u>Even or odd</u>

if (n&1) → odd

if (n&1==0) → even

1101
&1
0001 = = 1

------ --- 1 → odd

---------- 0 → even

(i) (N, i) check if ith bit is set or not in number N

43210
eg. 1 1 0 0 1 i = 3

mask: 01000
00 1000

ith

(mask & N != 0) → bit set

= 0 → not set

00001
1<<3
00 1000

third bit

000 - - 1 = 1_{10}

1<<3

if (N & (1<<i)) → set
else not set

2. Extract ith bit of number
Set the ith bit of a number

5 4 3 2 1 0
N = 1 1 0 0 1 0 i = 2
13
ans = (1 1 0 1 1 0)₂

mask = 1 << i

ans = mask | N

3. Clear the ith bit

mask = 1 << i (0 0 0 1 0 0 0 0

~mask = 1 1 1 1 0 1 1 1 1

ans = N & (~mask)

11
1
0 0
1 0 10

3. Remove the last set bit

110 1(1)0 → last set bit

testcase
10 = 1 0 1 0 only this is diff & all same
9 = 1 0 0 1
9&10 = 1 0 0 0
= 8

$\boxed{\text{ans} = n \& (n-1)}$

odd no → 1 1 0 1 → set bit
13 − 1 = 12
directly

even 1 0 0 0
1 0 1 0 0 0

4. check whether power of 2

100000 → n
011111 → (n−1)

edge case if(n==0) return false

$\boxed{\text{if}(n \& (n-1) == 0)}$ → power of two

except 0

5. count no of setbits in N

14 → 1110 ans = 3

```
cnt=0                          | while(n!=0)
while(n!=0){                   |     n=n&(n-1)
    if(n&1==1)                 |     cnt++;
        cnt++                  | }
    n=n>>1;                    | print(cnt)
}                              |
print(cnt);                    | O(set Bits)   15 1111
                               |                  O(4)
TC: O(Log N)                   |
    O(MSB)                     | same complexity
```

setabit — n|(1<<i)

check — n&(1<<i)

Q n integer is given
every integer appears twice two integer appears once

{1,1,2,5,3,2,3,4,7}

Soln: XOR of all element = 5^7
= 2

1st bit is zero	1st bit is set
1	2
5	2
4	7
4	3
	3
5	7

Sbka XOR

101
111
010 = 2

at this position only our no diff

odd one detector

⇓

2 possi...

⇓

2 numb...

```
XOR=0
for(i=0->n)
    xor ^= a[i];        } O(n)        TC: O(n)
}
```

other approches
1. Brute Force $O(n^2)$
2. map $O(n\log n)$

```
cnt=0;
while( xor){                     SC: O(1)
    if( XOR&1)
        break;                   O(32) -> O(1)
                                  think?
    cnt++;                        (1)
    XOR >>= 1;
}
XOR1 = 0      XOR2=0

for( i=0->n)
{                                 -> check this bit is set or not
  if(a[i] & (1<<cnt)){
      XOR1  ^= a[i]               O(n)
  }
  else
      XOR2 ^= a[i];
}
cout<<XOR1<<" "<<XOR2;
```

Q. Given n int, print XOR of all the subsets

arr[] = {1, 3, 2}   { } -> 0

Subsets = {1}, {3}, {2}, {1,3}, {1,2}, {3,2}, {1,3,2}

1 ^ 3 ^ 2 ^ 2 ^ 3 ^ 1 ^ 0 = 0
(ans)

dhyan se dekho --- !!

~~(1^3^2)^(3)^~~

$(\cancel{1})\wedge(\cancel{3})\wedge(\cancel{2})\wedge(\cancel{1}\wedge\cancel{3})\wedge(\cancel{1}\wedge\cancel{2})\wedge(\cancel{3}\wedge\cancel{2})\wedge(\cancel{1}\wedge\cancel{3}\wedge\cancel{2})$

count of each element is even
a^a=0

ans = 0

Ans is always zero

Generate all the subset

arr = {3, 2, 4}

0 → not take
1 → take

n = 3

no of subset = $2^3 = 8$

num	2	1	0 ← bit index	
0	0	0	0	{ }
1	0	0	1	{3}
2	0	1	0	{2}
3	0	1	1	{3, 2}
4	1	0	0	{4}
5	1	0	1	{3, 4}
6	1	1	0	{2, 4}
7	1	1	1	{3, 2, 4}

```
            (2^n - 1)
for( num = 0 → ((1<<n) - 1))
{
        vector<int> ds;
    for( bit = 0 → n-1).                  checkisset
        if( num & (1<<bit))
               ds.add( a[bit]);
    }
}

    for(auto it : ds)
          print(it);
```

Peter & combination Lock

method 1 → recursion call possibility. $T(2^n)$

powerset → $(n \cdot 2^n)$

② flag = 0
for(num = 0 → ((1<<n)−1))
        sum = 0
        for(bit = (0 → n−1)) {
                if(num & (1<<bits))
                        sum += a[i]
                else sum −= a[i];
        }
        if(sum % 360 == 0) flag = 1; break;
}
flag 1 → yes
0 → no

Bitmasking

Design a set data structure

1) add(x) → log(n) set $0 \leq x \leq 60$
2) removal(x) → log(n)
3) print all element → (ascending order)

focus

add(5)
add(1)
add(5)
add(3)
remove(5)

print = 1, 3

SC : O(1)
TC : 60(1) } by bit masking

long long x = 0

000 ---64 bits--- 000

→ O(1) operation

add(5) → (n | (1<<5))

n = 000--100000

n = 32

add(1) → n | (1<<1)

n = 000-100001

n = 33

add(3) (x | (1<<3))

n = ~~87~~41   101001

~~return~~

remove(5)   x & (1<<5)

set of bits if exist

x = 9   = 1001

~~print()~~ →

if doesn't exist then also remove

x & ~(1<<5)

clear ith bit

print()

code 2

```
for(bit = 0 → 60)
{
    if( x & (1<<bit))
        print(bit)
}
}
```

TC: O(1)

SC O(1)

add

add(n) → mask | (1<<n)

remove(n)

~~mask & n(1<<n)~~

mask & ~(1<<n) ✓

* highly used in DP

constrain : $0 < n < 60$