QPS

19 June 2023 19:10

'&', '|', '~', '^', '<<', '>>'

& → all 1→1

| → any 1→1

~ → 1→0

^ → odd no. 1→1

<< → ×2

>> → /2.

You are given an array of elements in which all numbers are appearing twice, except two numbers.

6, 8, 2, 3, 4, 3, 2, 6

8 ^ 4
1000
0100
1100 → 12

Find out those 2 numbers. :-

XOR of a numbers with itself is always '0'.

what does it signify??

011 → 3
101 → 5
^ 110 → 6.

7, 3
111
011
^ 100 = 4

011 → 3
101 → 5

6, 8, 2, 3, 4, 3, 2, 6

XOR = 12
1 1 0 0

8 (3rd)	4 (2nd)	2 (1)	1 (0)
0	1	1	0
0	0	1	1
0	0	1	0
1	0	0	0
0	1	0	0

6 → twice
3 → twice
2 → twice
8 → once
4 → once

of numbers which are not repeating ← 2nd & 3rd bit are diff.

→ I can make 2 clubs.

1st club
elements having 2nd bit set.
{6, 6, 4}
o/p = 4
↓ set variable

2nd club
elements having 2nd bit unset.
↓ unset
{3, 2, 3, 2, 8}
o/p = 8

2 nos. →

8 4 2 1
11 → 1 0 1 1
^ 3 → 0 0 1 1
1 0 0 0 → 8.

10 → 1 0 1 0
2 → 0 0 1 0
1 0 0 0 → 8

elements with 3rd bit set

unset.
{2, 3, 6, 2, 3, 6,
4

elements with bit set

8, → 8

2, 3, 6, 2, 3, 6, 4 → 4

For a particular i-th bit in binary, each no. has 2 choices, either that bit will be set or not → 0/1 (i-th)

{6, 8, 4, 3, 2, 3, 2, 6} → XOR = 12

2 / 3

// finding out last / first diff. bit. → because it is easier

→ You are given number (XOR) 12 find out its first set bit from right: (1100) → 2nd bit is set.

ct = 0, 1, 2 → set bit

```
while (num)
if (num & 1)
    break;
ct ++;
num = num >> 1;
```

1100
0011

ct = 2

mask = 1 << ct;

all 4 → 4.

mask ⇒ 00100 (00001)

1st bit
2 set.

```
setclub = 0
unsetclub = 0.
for (int i = 0; i < n; i++)
    if (mask & arr[i]) → bit is set.
        set ^= arr[i];
    else unset ^= arr[i];
cout << set << ' ' << unset << endl;
```

{6, 8, 3, 2, 4, 2, 3, 6.} → sort

2 2 3 3 4 6 6 8

cout << 4
cout << 8

n(n-1)/2

cout << 4
cout << 8

Bubble sort → $\frac{n(n-1)}{2}$
S ·
Insertion — at most $\frac{n(n-1)}{2}$

---

12 → (1100) → 1000
You have to make 1st (last from left)
(first bit from right) set bit 0.

i/p (12) → (8) o/p.
i/p (16) → (0) o/p.
16 = (10000) → 0.

4th bit

0001 0000
6 11101111
achieve if
4th bit → 0
rest → 1.

mask = 1 << 4
~(0001 0000) = 11101111

(12) → (8)
1100 1000

~0100
1100
1011
01000

~ 01100
00100
^ 01000
→ o/p

---

1 (001) → 0
2 → (010) → 0.
4 → (100) → 0
8 → (1000) →
16 → 7 0111
0
(16 & 15)

1 → 0
2 → 0
4 → 0
8 → 0
2 & 1
10
01
00

1 & 0 → 0.
001
000
000

4 & 3
100
011
000
8 & 7 →
8 = 1000
7 = 0111
0000

(12)
8 4 2 1
1 1 0 0
→ 0.
(4) → 0
4 & 3
→ 000

$2^2 - 3$
$(2^2 - 1) + \ldots$
8 + 4 0 0
8 + 3 0 0
→ (8)
12 & 11 → (8)

12 1100
11 1011
1000 → (8).

[n & (n-1)] → it will set
last set bit to 0.

8:40 PM → we will be back.

Prefix Array

arr = [1, 2, 3, 4, 5, 6]

5 times — Q[0, 4] → Find out the sum of elements from 0 to 4th index
5 — 1 + 2 + 3 + 4 + 5 = 15

4 times — Q[2, 5] → 3 + 4 + 5 + 6 → 18

3 times — Q[1, 3] → 9 = 2 + 3 + 4

6 times — Q[0, 5] → 21

array has → 10^6 elements
Queries → 10^6 queries

at most $10^6 \times 10^6$ → 10^{12} computations → TLE {Time limit Exceeded}

In C++, in 1 sec. you can do atmost $\boxed{10^8 \text{ computations}}$

---

arr = {1, 2, 3, 4, 5, 6}

→ pref.arr[] = {1, 3, 6, 10, 15, 21} ←

prefarr[0] → arr[0]
[1] → arr[1] + arr[0]
[2] → arr(1) + arr[0] + arr(2)
[3] → + arr[3]

Q [0, 5] → prefarr[5] − pref[0 − 1]
21 — −1 ✗

Q [1, 5] → 20
pc[5] − pc[1 − 1]
21 − 1 = 20

arr = [1, 2, 3, 4, 5, 6]
pref arr = [1, 3, 6, 10, 15, 21]

pref arr[0] = arr[0]

for (int i = 1 ; i < n ; i++)
{ pref.arr[i] = arr[i] + prefarr[i−1];
}

n times
if 10^6 elements are given → 10^6 times

i = 2, 3

10^6 Queries are given

2×10^6 computations

3×10^6 computation $< 10^8$

for 1 query $[i, j]$

$pa[j] - pa[i-1]$

1 computation

1 computation

If more than 1 zero → all zeroes

(n) → {0};

print (arr)

→ if there is only 1 zero / no zeroes

at the index of 0 prod. of rest of the array will come 0 at '0' will come rest of the array

10

prod of arr = 1 → prod. of whole array

for (int i=0; i<n; i++)

cout << prod of arr/arr[i] << endl

i^{th} → from 0 to n-1

0 1 n-1

		----		....	

(1) prod (0, i-1) x prod (i+1, n-1)

(n) prod (1, n-1)

i = 0 to n-1

ans [i] = prod (0, i-1) x prod (i+1, n-1)

1 computation

x

10^6 → 10^6 computations

ans [i] = prod [0 to i-1] x prod [i+1, n-1]

i = 0 prod [0, -1] 1 — [1, n-1]

i = 1 prod [0] x — [2, n-1]

i = 2 p — [0,1] x — (3, n-1)

i = 3 [0,1,2] x — [4, n-1]

⋮

i = n-2 [0, n-3] x — [n-1]

i = n-1 [0, n-2] x p [n, n-1] (1)

a(0)xa(1) a[0] ... a[3]

a[0]xa(1)xa(2)

left prod prefix :- [1 | a[0] | | | |]

leftprodprefix :-

| 1 | a[0] | | | |

a[0] x a[1] x a[...

rightprodprefix :-

a[n-1]

a[n-2] x a[n-1]

a[n-3] ... a[n-1]

a[1] x a[2] ... a[n-1]