BIT Manipulation - 2

Hello Everyone

Lost Packet Identification:

Scenario:
In a large data center, servers communicate with each other and with the outside world by sending and receiving packets of data. Each packet has a unique identifier (ID), and due to the nature of transmission protocols, every packet is supposed to be sent and then acknowledged (ACK) by the receiving end. In an ideal scenario, for every packet ID, there should be exactly two occurrences in the network log: one for the packet being sent and another for its acknowledgment.

Problem:
Due to network issues, a packet got lost. Now we need to find out which packet got lost.

You are given an array with packet IDs from sender and receiver ends, only one packet was lost and hence it occurs only once in the array. Find this element.

Example -
`{ 4, 1, 2, 4, 2, 1, 5 }`
5 is the answer since it is present only once.

The above problem is known by the name Single Number, let's see it further.

Q) We are given an integer array where every number occurs twice except for one number which occurs just once. find the number.

Eg. {4, 5, 5, 4, 1, 6, 6}

Eg. {7, 5, 5, 1, 7, 6, 1, 6, 4}

Quiz 1 : 120 ^ 5 ^ 6 ^ 6 ^ 120 ^ 5

Approach :

Since A ^ A = 0, taking a XOR of all elements will cancel out all paired elements and give the single element.

3 ^ 4 ^ 3 ^ 4 ^ 2

↳ 3 ^ 3 ^ 4 ^ 4 ^ 2

Code:

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

{3,4,3,4,2}

i

ans = 0 ^ 3 ^ 4 ^ 3 ^ 4 ^ 2

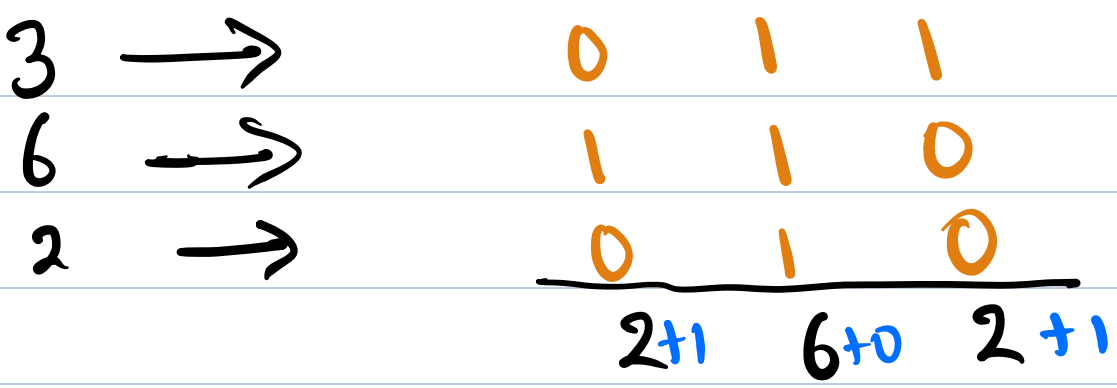
TC: O(n)

i=0	0^3	→ 3	i=3	4^4	→ 0
i=1	3^4	→ 7	i=4	0^2	→ 2
i=2	7^3	→ 4			

Approach 2:

Eg. {2, 3, 5, 6, 3, 6, 2}

		2	1	0
2	→	0	1	0
3	→	0	1	1
5	→	1	0	1
6	→	1	1	0



Count of set bit at index i

→ Even (the index of ans is 0)

→ odd (the index of ans is set.)

1) Iterate on all bits one by one.
2) Take the count of number for which that bit is set.

If count = 2x+1 (odd) → set that corresponding bit in answer.

Else

Unset that bit in ans.

or Do nothing

Code:

```
ans = 0;
for (i = 0; i < 32; i++)
{
    cnt = 0;
    for (j = 0; j < n; j++)
    {
        if (checkBit(A[j], i) == true)
        {
            cnt++;
        }
    }
    if (cnt % 2 == 1)
    {
        ans = ans | (1 << i);
    }
}
return ans;
```

{2, 2, 2, 6, 6, 8}

3 2 1 0

2 → 0 0 1 0

2 → 0 0 1 0

ans = 0

6 → 0 1 1 0

6 → 0 1 1 0

8 → 1 0 0 0

i = 0,	cnt = 0	ans = 0
i = 1,	cnt = ~~0~~ 4	ans = 0
i = 2,	cnt = ~~0~~ 2	ans = 0
i = 3,	cnt = ~~0~~ 1	ans = 8

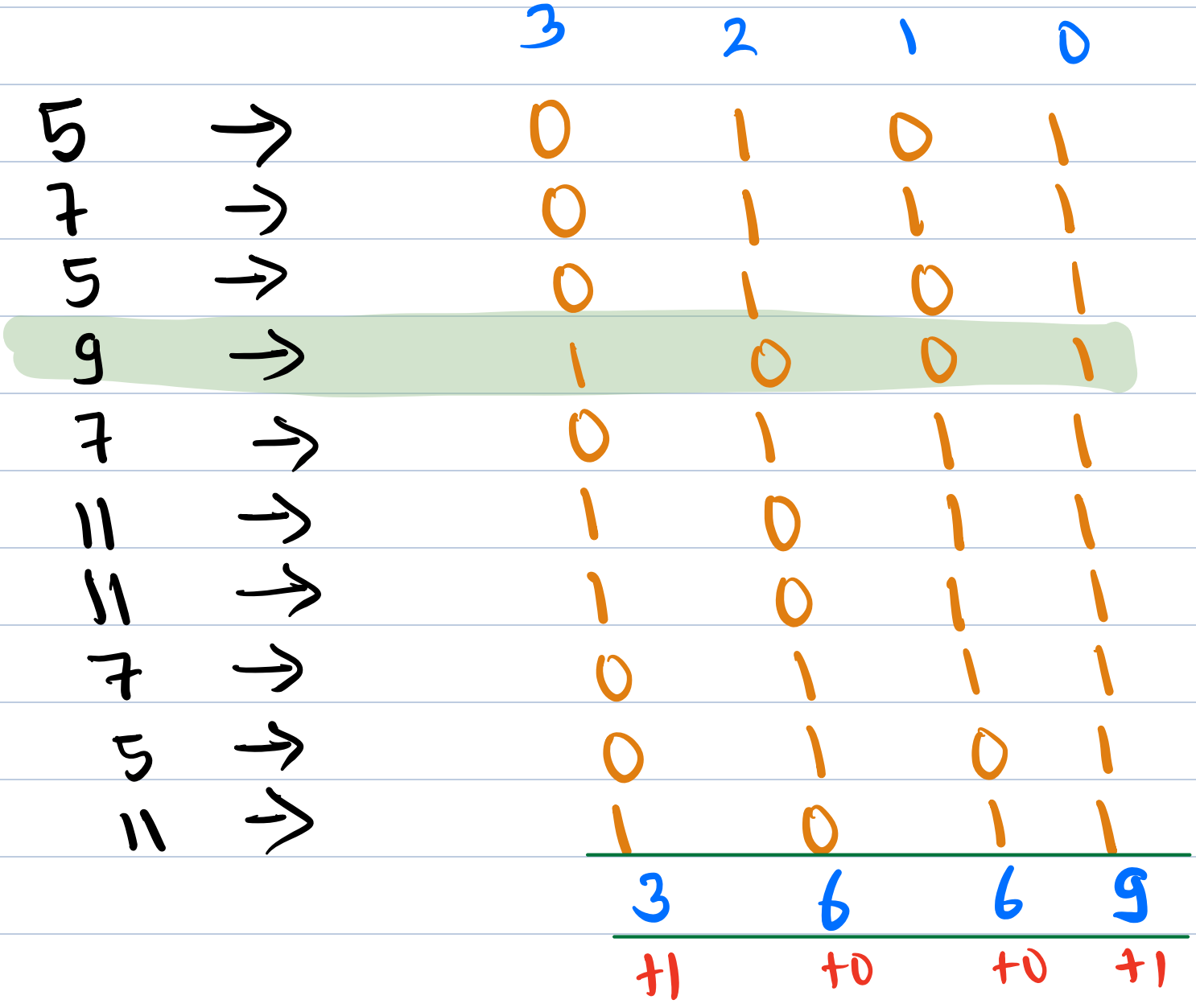
TC: O(32 * n)

O(n)

9) Variation to above problem.

All numbers exist in triplet except one. find that number.

Eg. {5, 7, 5, 9, 7, 11, 11, 7, 5, 11}



1) Iterate over all bits one by one.

2) Take the count of number for which that bit is set.

If count = 3x+1 → set that corresponding bit in ans.

Else → unset that bit in ans OR Do nothing.

```
ans = 0
for (i = 0; i < 32; i++)
{

    cnt = 0
    for (j = 0; j < N; j++)
    {
        if (checkBit(A[j], i) == true)
        {
            cnt ++;
        }
    }

    if (cnt % 3 == 1)
    {
        ans = ans | (1 << i);
    }
}

return ans;
```

Q) Given an integer array, all the elements will occur twice except two. Find those two elements.

Eg. {4, 5, 4, 1, 6, 6, 5, 2}

Step 1: Take XOR of all elements.

4 ^ 5 ^ 4 ^ 1 ^ 6 ^ 6 ^ 5 ^ 2

= 1 ^ 2

= 3

```
 1  0
 0  1
 1  0
-----
 1  1   → 3
```

Group 1
0th bit is set
5, 1, 5

Group 2
0th bit is unset
4, 4, 6, 6, 2

Group 1
1st bit is set

6, 6, 2

Group 2
1st bit is unset

4, 5, 4, 1
5

{ 4, 5, 4, 1, 6, 6, 5, 2 }

	2	1	0
4 →	1	0	0
5 →	1	0	1
1 →	0	0	1
6 →	1	1	0
2 →	0	1	0

{ 4, 5, 4, 1, 6, 6, 5, 9 }

4 ^ 4 ^ 5 ^ 5 ^ 6 ^ 6 ^ 1 ^ 9

$_{3\,2\,1\,0}$

0 0 0 1
1 0 0 1
---
1 0 0 0 → 8

Group 1
If 3rd bit is set

9

Group 2
If 3rd bit is unset -

4, 5, 4, 1, 6
6, 5

4 ^ 5 ^ 4 ^ 1
^ 6 ^ 6 ^ 5

	3	2	1	0
4 →	0	1	0	0
5 →	0	1	0	1
6 →	0	1	1	0
1 →	0	0	0	1
9 →	1	0	0	1

9

Code:

```
XORAll = 0;
for(i=0; i<N; i++)
{
    XORAll = XORAll ^ A[i];
}
Pos = 0;
for(i=0; i<32; i++)
{
    if (checkBit(XORAll, i)==1)
    {
        Pos = i;
        break;
    }
}
```

O(N)

O(32)

```
Group1 = 0
Group2 = 0
```

O(N)

```
for (i=0; i<N; i++)
{
  if (checkBit(A[i], pos) == true)
  {
    group1 = group1 ^ A[i];
  }
  else
  {
    group2 = group2 ^ A[i];
  }
}

print (group1);
print (group2);
```

TC: O(N)

Break: 10.35 pm

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

1 ^ 1 ^ 2 ^ 3 ^ 4 ^ 2 ^ 5 ^ 4

= 3 ^ 5

```
      2 1 0
      0 1 1
XOR   1 0 1
-------------
    1 1 0  => 6
```

Group 1
1st bit is set
2, 3, 2

Group 2
1st bit is unset
1, 1, 4, 5, 4

```
        L
1 -   0 0 1
2 -   0 1 0
3 -   0 1 1
4 -   1 0 0
5 -   1 0 1
```

Q) Given N array elements, choose two indices (i, j) such that (arr[i] & arr[j]) is maximum and return that value.

0 1 2 3 4

Eg. { 5, 4, 6, 8, 5 }

```
  100
& 110
  ---
  100 -> 4
```

```
  0110
& 1000
  ----
  0000 -> 0
```

```
  101
& 101
  ---
  101 -> 5
```

Quiz: { 21, 18, 24, 17 }

```
21 ->  1 0 1 0 1
18 ->  1 0 0 1 0
24 ->  1 1 0 0 0
17 ->  1 0 0 0 1
```

24 & 21

```
  11000
```

17 & 18

```
  10001
& 10010
  -----
```

10101
10000 → 16

21 & 18 → 10101 & 10010 = 10000 → 16

21 & 17 → 10101 & 10001 = 10001 → 17

Bruteforce: Using two for loops and calculating bitwise & for all possible pairs and storing the maximum of all them.

Observation

1) The bit will be 1 if bits of both the number is 1.

2) To maximize the answer, we should try to set bit to the maximum left (MSB).

3) Iterate from MSB.

Dry Run:

Eg. { 26, 13, 23, 28, 27, 7, 25 }

0 1 2 3 4 5 6

i

	31 5	4	3	2	1	0
26 →	0 0	1	1	0	1	0
13 →	0 0	0	1	1	0	1
23 →	0 0	1	0	1	1	1
28 →	0 0	1	1	1	0	0
27 →	0 0	1	1	0	1	1
7 →	0 0	0	0	1	1	1
25 →	0 0	1	1	0	0	1

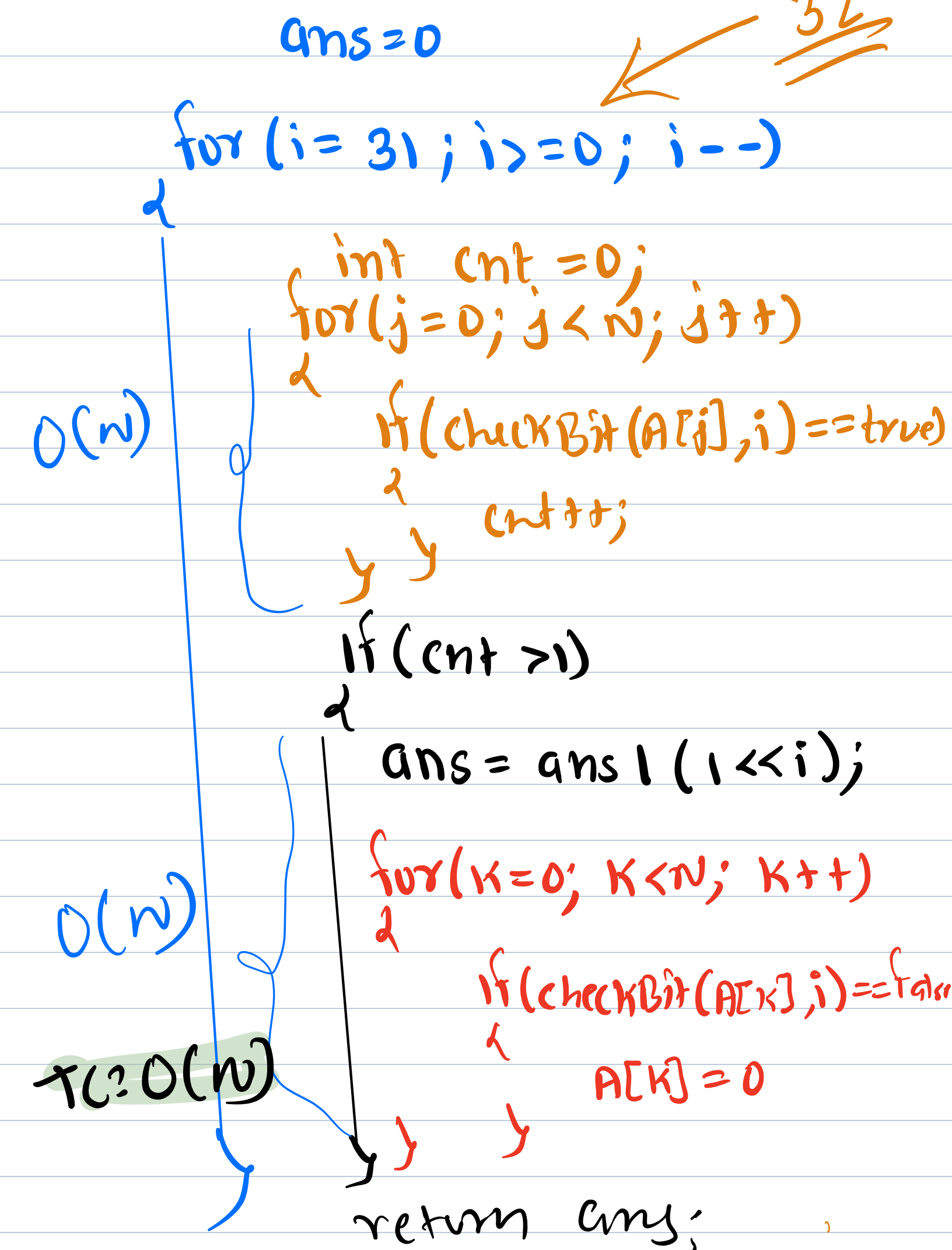
```
  0 1 1 0 1 1
  0 1 1 0 1 0
-------------
  0 1 1 0 1 0  → 26
```

ans = 0; cnt = 1

i

	31 ... 5	4	3	2	1	0
26 →	0 0	1	1	0	1	0
~~13~~ 0 →	0 0	0	0	0	0	0
~~23~~ 0 →	0 0	1	0	0	0	0
~~28~~ 0 →	0 0	1	1	1	0	0
27 →	0 0	1	1	0	1	1
~~7~~ 0 →	0 0	0	0	0	0	0
~~25~~ 0 →	0 0	1	1	0	0	0
Ans->	0 0	1	1	0	1	0

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


ans=0
32
for(i=31; i>=0; i--)
int cnt=0;
for(j=0; j<N; j++)
if(checkBit(A[j],i)==true)
cnt++;
O(N)
if(cnt>1)
ans = ans | (1<<i);
for(k=0; k<N; k++)
if(checkBit(A[k],i)==fals
A[k]=0
O(N)
TC: O(N)
return ans;

Q) Calculate the count of pairs for which bitwise & is maximum.

0	1	2	3	4	5	6	7	8

Eg. {5, 4, 3, 4, 2, 1, 5, 4, 5}

0, 6
6, 8
0, 8

$\Rightarrow \frac{N*(N-1)}{2}$

(5) $\rightarrow \frac{5*(5-1)}{2}$

$= \frac{5*4}{2} = 10$

H.W