PROBLEM-SOLVING-SESSION

PRE QUES

a = 101101

b = 100001

BITWISE OR 101101

BITWISE OR

1	1	1
0	1	1
1	0	1
0	0	0

a = 101] = 1

b = 100]

c = 101] = 111 ≠ 1

1 0 11

BITWISE OR ⇒ If any of the bit is 1 the the BITWISE OR is 1

CONCLUSION

PRE-2

arr : 0 1 2 3 4 - - N-1

N integers array

1) what do you mean by subarray ?

2) how many total subarray are possible from array of N integers ? N*(N+1)/2

arr : 0 1 2 3 4 5 6 7

array size = 8

= 1 + 2 + 3 + ---- 8 (N terms)

= 8 (8+1)

2 $\qquad\qquad$ $\frac{N * (N+1)}{2}$

QUES Given an array A of N integers with elements 1 & 0.

Number of subarrays with BITWISE OR as 1.

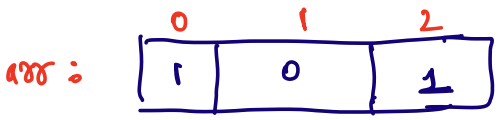
arr: [1 0 1]

[1] = 1 ✓ $\qquad$ [1,0] = 1 ✓ $\qquad$ Ans = 5

[0] = 0 $\qquad$ [0,1] = 1 ✓

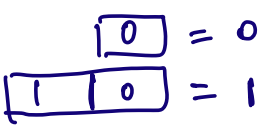
[1] = 1 ✓ $\qquad$ [1,0,1] = 1 ✓

Solution

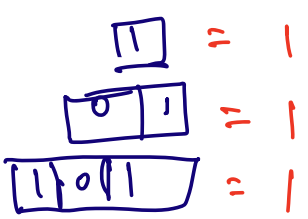


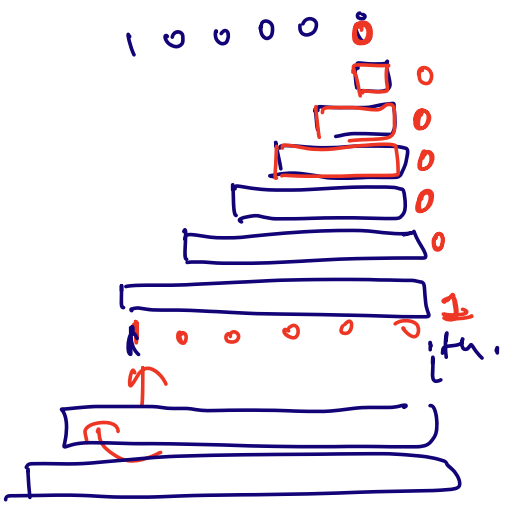
ending with 0^{th} indx : 1

ending with 1^{st} indx : 1

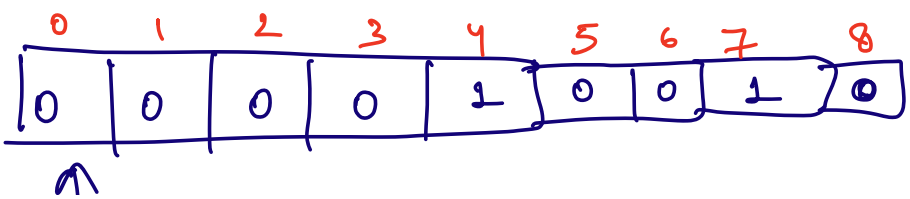


ending with 2^{nd} index = 3





take care about last one.

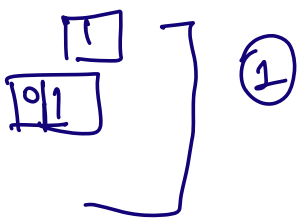


ending 0^{th} index = [0)

ending with 3^{rd} index = there is no 1's before this index

= 0

ending with 4^{th} index = 5



CODE:

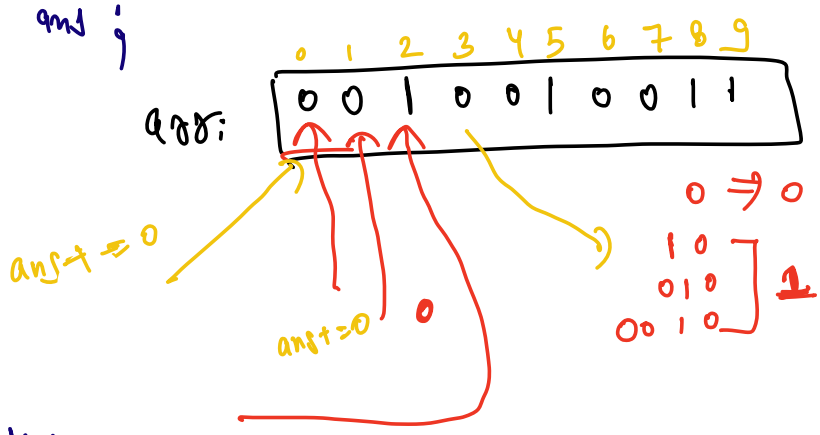
```
long solve ( vector<int> A)
{
    int last_one = 0 ; // position of last_one present.
    long ans = 0 ;

    for ( int i=0 ; i<n ; i++)
    {
        if (B[i] == 1) last_one = i+1 ;
        ans += last ;
    }
```

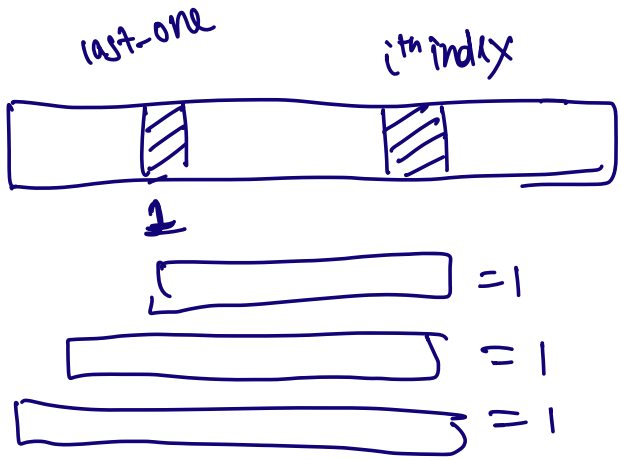
Time-complexity = O(N)

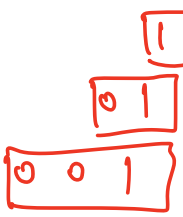
return ans ;

space = O(1)

3







last_one = 3

ans += last_one.

ans += 3

last_one = 3

ans = 3

QUES-2

PRE-REQ-1

{1, 2, 3} = subsets

{} {1} {2} {3}

{1,2} {2,3} {1,3}

3 size

{1,2,3} = 8

{1,2} = {}

2 size {1} {2}

{1,2} = 4

$\{1, 2, 3, 4\} =$

$\{1\} = 2^1$

$\{1,2\} = 2^2$

$\{1,2,3\} = 2^3$

$\{1,2,3,4\} = 2^4$

$\{1\} = 2^1$

$\{1,2\} = 4 = 2^2$

$\{1,2,3\} = 8 = 2^3$

$\{1,2,3,4\} = 16 = 2^4$

$\{1, 2, \ldots N\} = 2^N$

$\{1, 2, 3 \ldots 13\} = 2^{13} = 8192$

Total no. of subsets.

QUES Find the N^{th} Magic Number

Magic Number : A number that can be expressed as a power of 5 or sum of unique powers of 5

First Magic Numbers : 5, 25, 30, 125, 130 ...

5^1 5^2 (5^1+5^2) (5^3+5^1)

(5^3)

3^{rd} magic number = 30

$\Rightarrow (5^1, 5^2, 5^3)$

$\Rightarrow \{5\} = 5$

$= \{6^2\} = 25$

$= (5^3) = 125$

A^{th} Magic Number

$(1 <= A <= 5000)$

$(5^1, 5^2) = \{5^1+5^2\} = 30$

$(5^2, 5^3) = 25+125 = 150$

$(5^1, 5^3) = 130$

$(5^1, 5^2, 5^3) = 5+25+125 = 155$

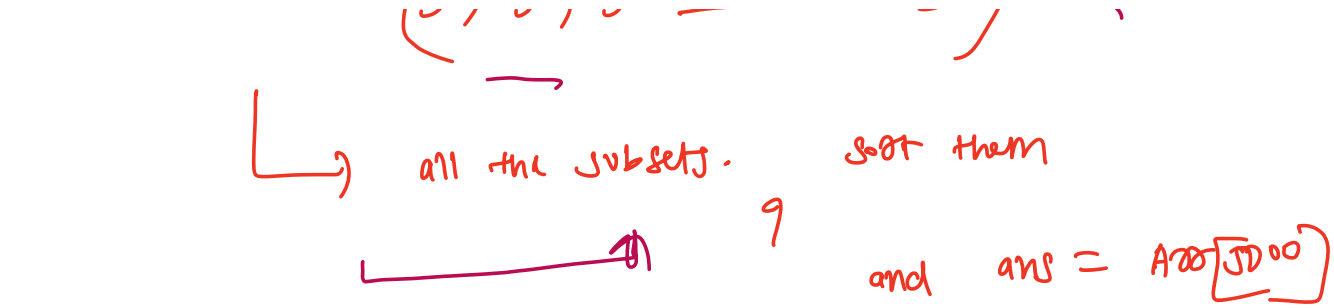
$\{5^1, 5^2, 5^3, 5^4\} \Rightarrow$ 15 Magic Number.

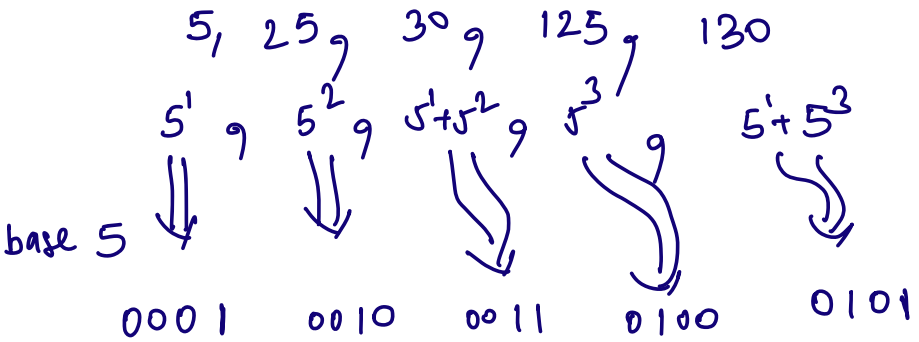
$\{5^1, 5^2, 5^3 \_\_\_ 5^{13}\} \Rightarrow$ 5000^{th} Magic Number

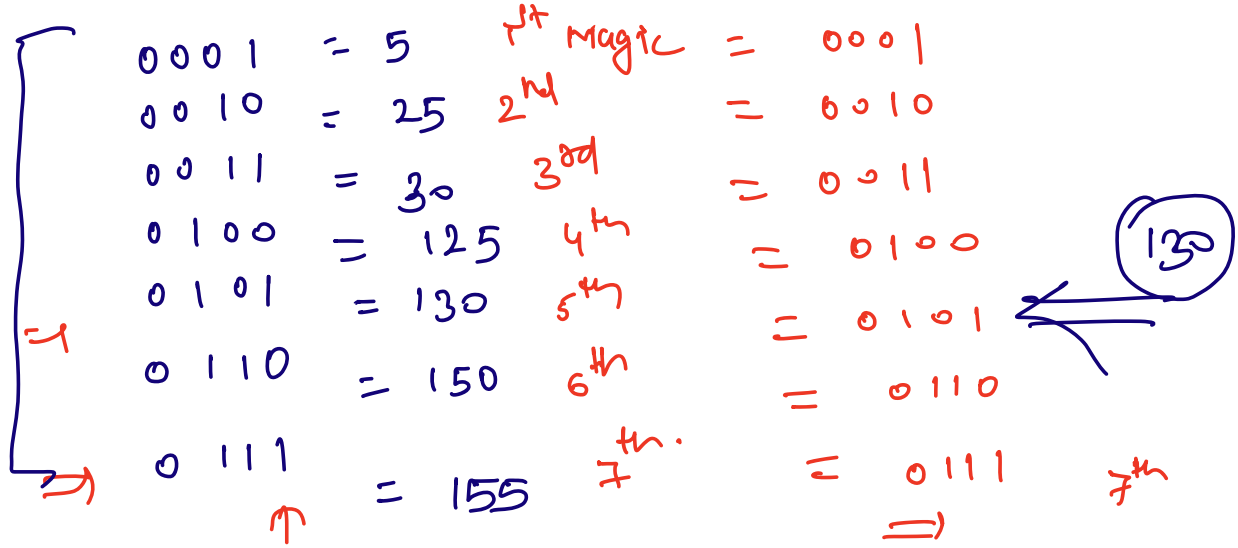
$\hookrightarrow$ 8192 Numbers (Subsets)

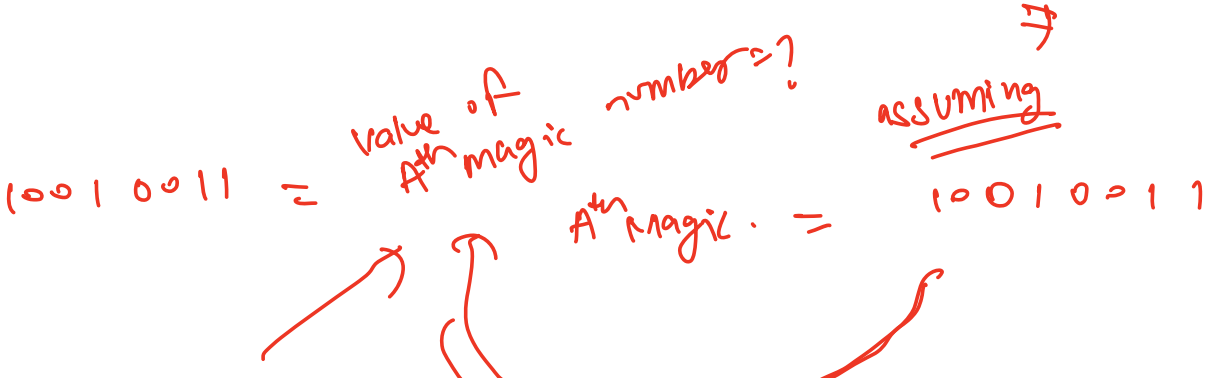
all the subsets.

$(5^1, 5^2, 5^3 \_\_\_\_ 5^{13}) \Leftarrow$


all the subsets.
sort them
and ans = ARR[3]


5, 25, 30, 125, 130
5^1
5^2
5^1+5^2
5^3
5^1+5^3
base 5
0001
0010
0011
0100
0101


0001 = 5
0010 = 25
0011 = 30
0100 = 125
0101 = 130
0110 = 150
0111 = 155
1st Magic = 0001
2nd = 0010
3rd = 0011
4th = 0100
5th = 0101
6th = 0110
7th. = 0111
7th
130


10010011 = value of Ath magic number =?
Ath Magic. =
assuming
7
10010011

Convert this into base 5 then A^{th} magic number value.

Code:

```
int Ath magic number (int A)

    int ans = 0;        int pow_5 = 5;

    for (int i = 0; i < 32; i++)
    {
        if (A & (1<<i) == 1)        // whether ith bit is set or not
        {
            ans += pow_5;
            pow_5 = pow_5 * 5;
        }
    }
    return ans;
```

pow(5, i) ⇒ pow(5, i+1)

T.C ⇒ O(32)

S.C ⇒ O(1)

A^{th} Magic ⇒ 0 0 0 1 0 1 ≡ 5

1 0 0 1 0 1 1

↑ ith

0th bit

$= 5^0$

5^1

2

5
x

5 ⇒ $125 + 5 = 130$

ROW TO COL ZERO

2-D integer matrix A. (positive)

Make all the element in the row or col. zero (0)
if $A[i][j] == 0$

Make the entire row & col to zero

Mat :
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 9 & 2 & 0 & 4 \end{bmatrix}$$

return Matrix
$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

row ⇒
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \text{col: } \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 9 & 6 & 7 & 0 \\ 9 & 2 & 0 & 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 \\ x & x & x & 0 \\ x & x & 0 & x \end{bmatrix}$$

⇓

n = -1

$$\begin{bmatrix} 1 & 2 & x & x \\ x & x & x & 0 \\ x & x & 0 & x \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

code:

N = no. of rows M = columns.

```
for ( int i=0 ; i<n ; i++)
    int flag = 0
    for ( int j=0 ; j<m ; j++)
    {
        if (A[i][j] == 0)  flag = 1
    }
    if (flag == 1)  for( int j=0 ⟶ m )
                        if (A[i][j] != 0)
                            A[i][j] = -1;
}
```

0 is present

columns

```
for (int i → n)
    for (j → (0, m)) {
        if (A[i][j] == -1)
            A[i][j] = 0;
    }
return A;
```

TC = O(N+M)

SC = O(1)

QUES

PRE-REQ

Array of N integers.

(*) answer always exists.

majority element

more than $\lfloor N/2 \rfloor$ times.

arr: +1 +1 -1 -1 +1

[6 6 5 3 6]

N = 5

N/2 = 2

element

ans = 6

arr: [3, 2, 5, 3 3] ans = 3

arr: [2 2 2 2 1 1 1] ans = 2

is there possibility to have 2 majority elements?
if yes array = ?

OR always 1 Majority element

arr : [2 2 2 2 1 1 1]

-1 -1 -1

+1 +1 +1 +1

=

arr : [6 6 5 3 6]

Maj = 6

Maj freq = 1

Maj freq = 2

Maj 6 freq = 1

Maj 6 freq 0

TC O(N)

SC = O(1)

Ques N/3 repeat number

array of N integer.

value which occurs more than N/3

arr: [3 2 3] n=3 $3/3 = 1$ times.

Ans = 3

arr: [4 4 2 1 4] = ? 4

arr: [2 3 2 3 5 6] = ? = No majority

maj1 maj2 $6/3$? 2 times.

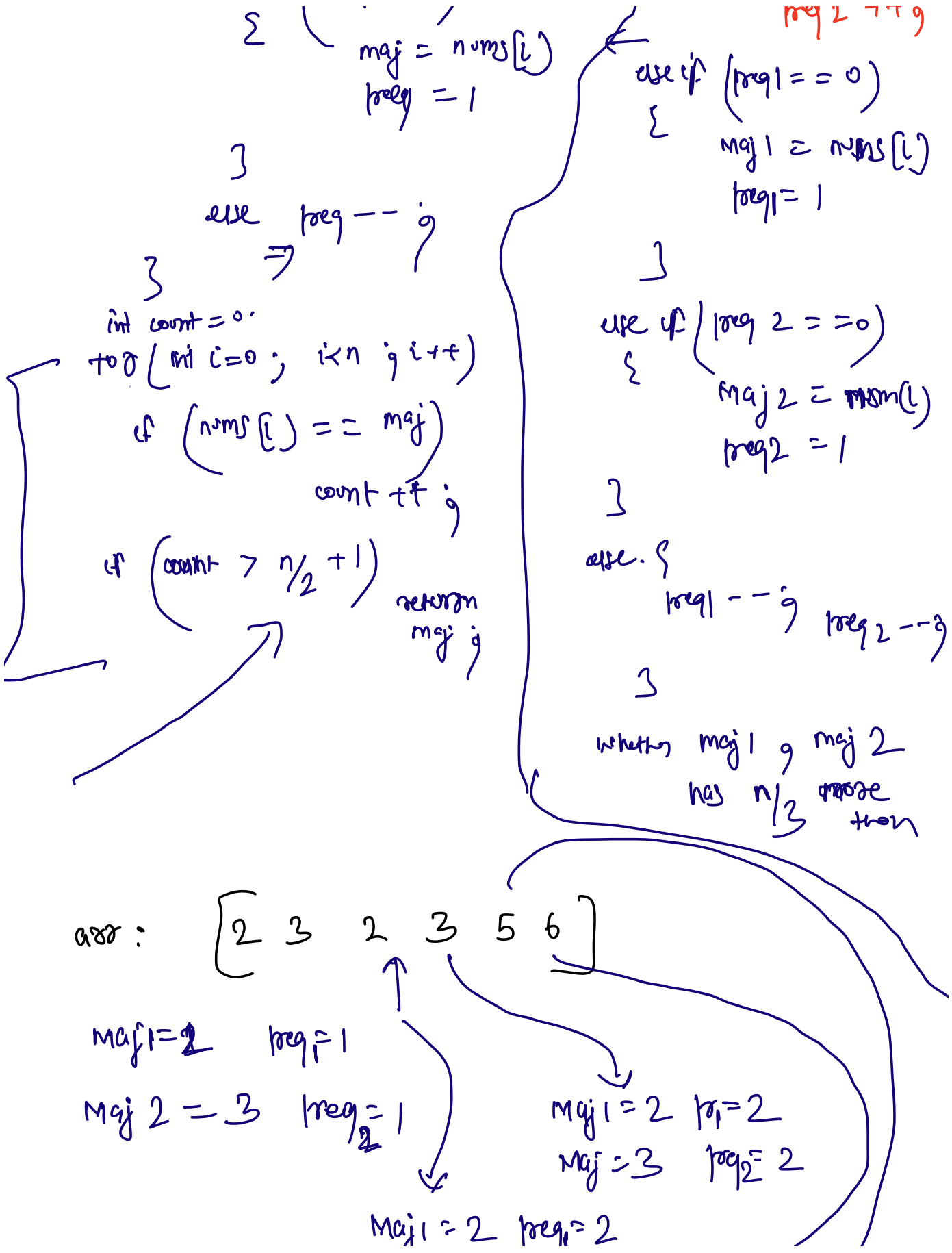
(FK) at most 2 majority elements are possible

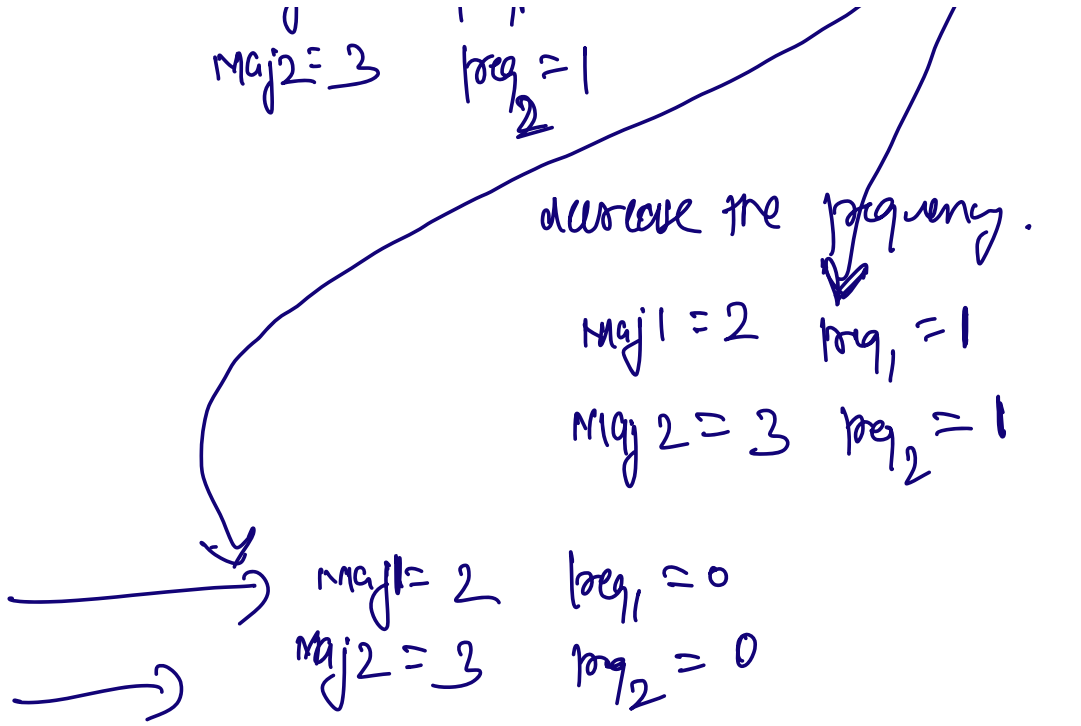
$n/2$ repeat number

```
int Maj( vector<int> nums)
{
    int Maj = 0 ;
    int freq = 0 ;
    for( int i=0 ; i<n ; i++)
    {
        if (nums[i] == Maj)
            freq ++ ;
        else if ( freq == 0)
```

$n/3$ repeat numbs.

```
INT Majf ( vector<int> nums)
    int Maj1 = 0 ; int freq1=0
    int Maj2 = 0 ; int freq2=0
for (int i=0 ; i<n ; i++)
{
    (if nums[i] == maj1)
        freq1 ++ ;
    elsif (nums[i] == maj2)
```


maj = nums[i]
freq = 1
else freq--;
int count = 0;
for (int i=0; i<n; i++)
if (nums[i] == maj)
count++;
if (count > n/2 + 1)
return maj;
freq2 ++;
else if (freq1 == 0)
maj1 = nums[i]
freq1 = 1
else if (freq2 == 0)
maj2 = nums[i]
freq2 = 1
else {
freq1--; freq2--;
Whether maj1 & maj2 has n/3 more than
arr: [2 3 2 3 5 6]
Maj1=2 freq1=1
Maj2=3 freq2=1
Maj1=2 freq1=2
Maj=3 freq2=2
Maj1=2 freq1=2



N/3 repeat = O(N) Time complexity

= O(1) SPC

arr: [1 1 1 2 2 2]

N/3 = 2

maj1 = 1

maj2 = 2

arr: [1, 2] N/3 = 0

maj1 = 1

maj = 2