Homework discussions.

Agenda.

(i) Excel column number ✓
(ii) Do rectangles overlap
(iii) Repeating and missing element in the array.
(iv) Reverse bits
(v) Single number 3.

(i) Excel column number.

i/p : String of characters from A-Z. [Capital].

A, B, C

AA → 27
AB → 28
AZ → 52.
BA → 53
BB → 54

A → 1 ✓
B → 2 ✓
C → 3 ✓
:
Z → 26 ✓

26^1 26^0

A A → $26^0 * A + 26^1 * A = 1 + 26 = 27$.

A B

A Z → $26^0 * Z + 26^1 * A = 26 + 26 = 52$.

Soln.

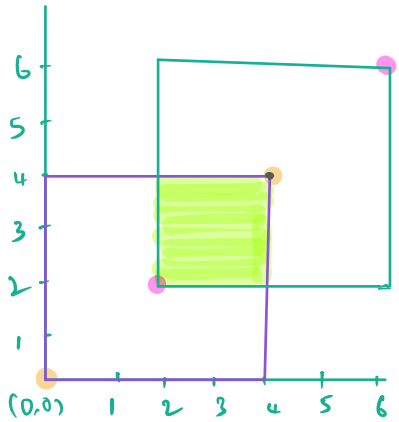
```
int stringToNumber(String A)
{
    int mul=1; int value=0;
    for(int i= A.length()-1; i>=0; i--)
    {
        char ch= A.charAt(i);
        value = value + (mul*(ch-'A'+1));
        mul = mul*26;
    }
    return value;
```

}

2) Do rectangles overlap. → [Yes, NO]

Four points. (A,B) (C,D) (E,F), (G,H).

↙ (0,0), (4,4), (2,2), (6,6).



max (A, E) → xs → (2)

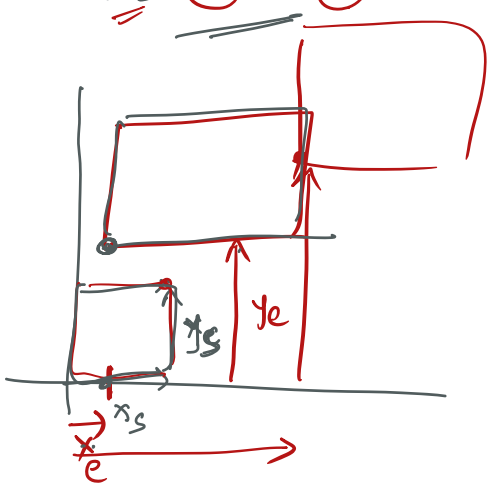
max (B, F) → xe → (2).

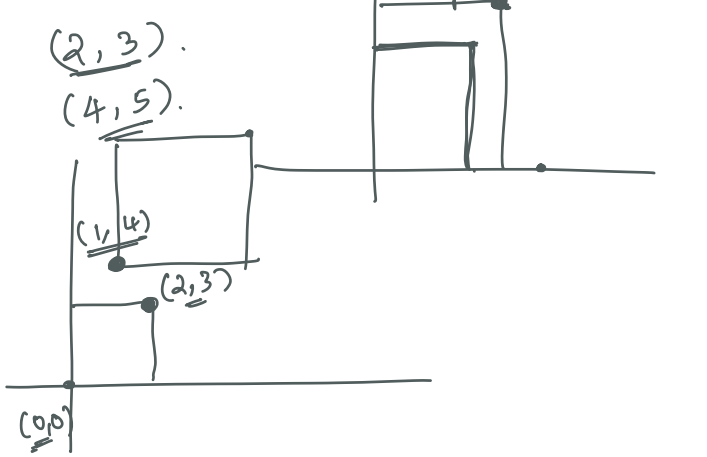
min (C, G) → ys → (4)

min (D, H) → ye → (4).

(2,2) & (4,4).

$(x_s < x_e \;\&\&\; y_s < y_e)$ =





(i) int doRectanglesOverlap(int A, int B, int C, int D, int E, int F, int G, int H)

{

int x_s = Math.max(A,E), x_e = Math.max(C,G)

int y_s = Math.min(C, G), y_e = Math.min(D, H)

if($x_s < x_e$ && $y_s < y_e$)

return 1;

else

return 0;

}

3) Repeating and missing number. (+ve no's). in 1 - 5. (1 - to N)

i/p : [3 1 2 5 3]. ← 5.
(indices 0 1 2 3 4)

missing → 4
repeating → 3.

1 to N.

→ [0 1 1 1 0 1] ←
(indices 0 1 2 3 4 5)

→ [... (4).

→ [1, 2, 3, ..., 98, 99, 98, ...] ← 99 elements.

97

(N+1)

[0 1 ... 1 1]
0 1 2 3 --- 98 99.

TC: O(N).

SC: O(N). → O(1).

(indices 0 1 2 (3) 4)

i/p → [(3) 1 2 (5) 3] ←

-5

[(1)] ← O(n).
(indices 0 1 2 3 4 5)

(indices 0 1 2 (3) 4)

[3 1 2 5 3].

[3 1 2 -5 3]

[3 -1 2 -5 3].

[3 -1 -2 -5 (3)].

[ ]-4

(indices 0 1 2 3 4)

[3 1 2 5 3].

3 → 2.

→ [3 1 -2 5 3].

1 → 0.

→ [-3 1 -2 5 3]

|-2| = 2. → 1

→ [-3 -1 -2 5 3].

5 → 4.

→ [-3 -1 (-2) 5 -3].

|-3| ← 3 → 2.

→ [-3 -1 -2 5 -3] ← ← ~~missi~~ repeating ← 3.
(indices 0 1 2 (3). 4.)

3+1

→ [11, 12, 13, 13, 15] ←

Code:

1 to N.

0 , 0 :

```
int[] findMissingAndRepeating (int[] arr,
                               int size)
{
    int missingNo = -1;
    int repeatingNo = -1;
    for (int i=0; i<size; i++)
    {
        int abs_val = Math.abs (arr[i]);
        if ( arr(abs_val -1) >0)
        {
            arr[abs_val -1) = -arr (abs_val -1);
        }
        else
        {
            repeatingNumber = abs_val;
        }
    }
    for (int i=0; i<size; i++)
    {
        if(arr[i] >0)
        {
            missingNumber = i+1;
        }
    }
    return missing & repeating;
}
```

7:09.

4) Reverse bits.

48 → _ _ _ _ _ _ ← res.

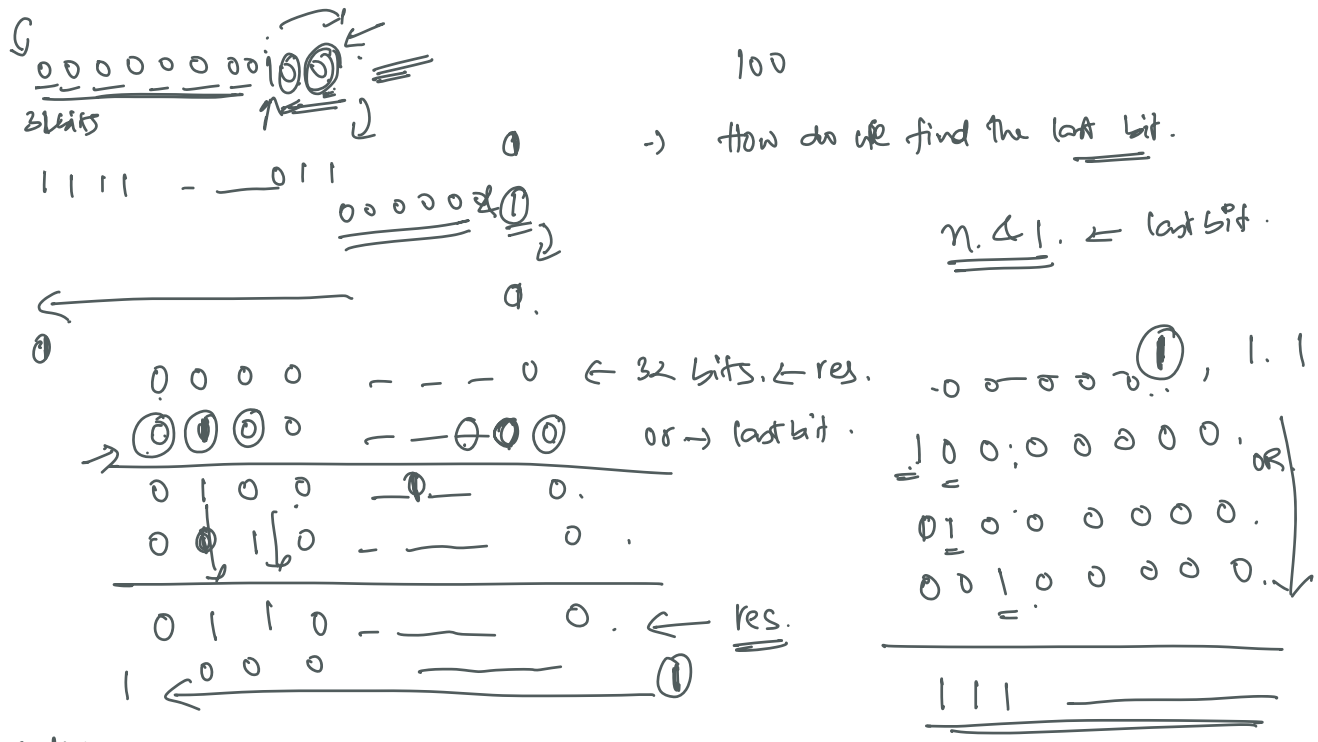
→ 10010.

→ 01101

32 ← bit.

~ ← tilde.

000 000 (1)



Code.

```
long reversebits (long number)
{
    long res=0;
    for (int i=0; i<32; i++)
    {
        long lastBit = (number>>i)&1;
        res = res | (lastBit<<(31-i));

    }
    return res;

}
```

5). Single number 3.

Two no's they appear once.

i/p: [3 4 3 4 8 4 4 32 7 7]

find the no's which appear only once.

XOR.

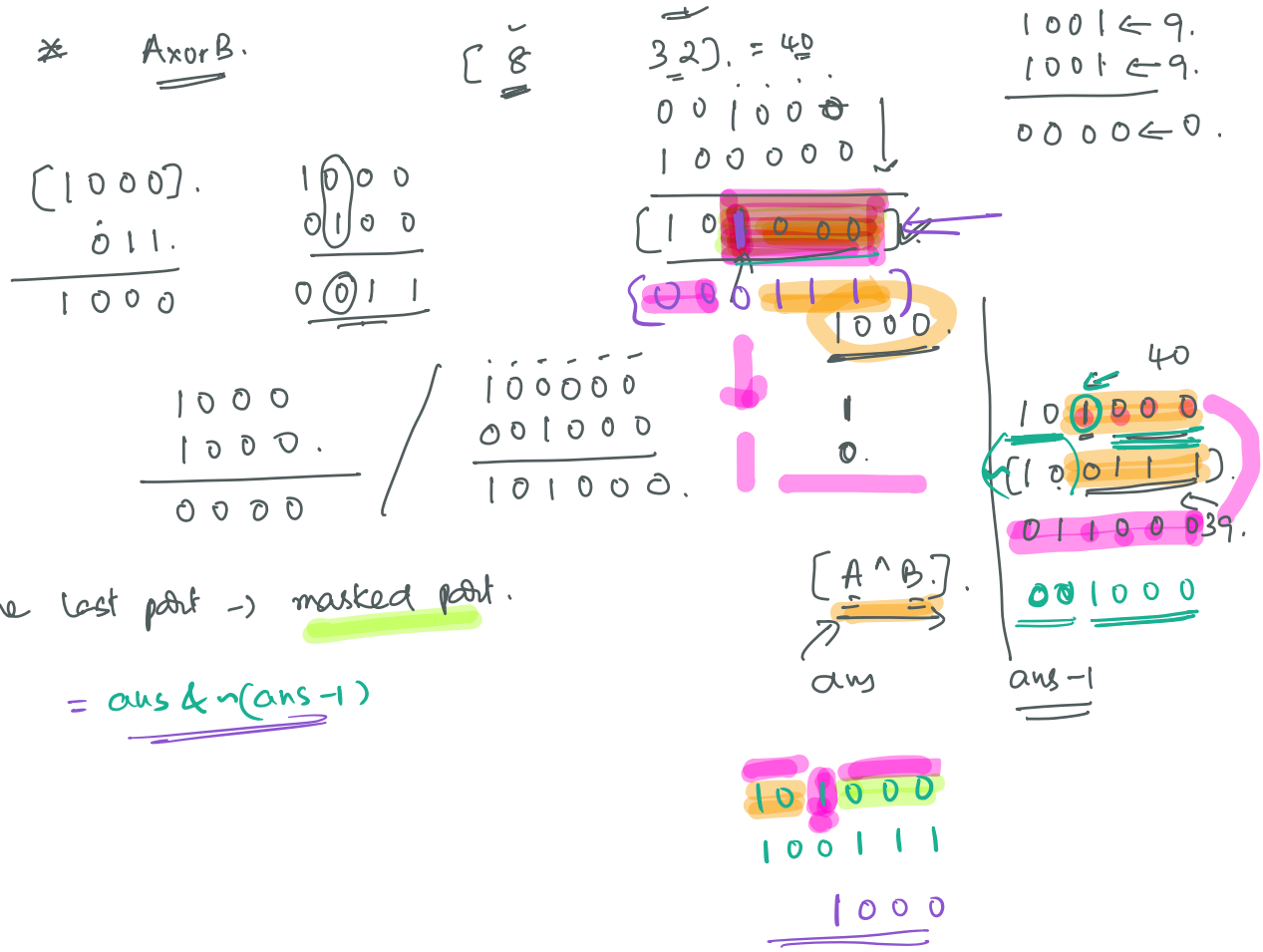
1 0 -> 1

0 1

0 0 -> 0.

1 1

Pre-requisite. i/p : [4 4 3 3 2 2 5].



The last part -> masked part.

= ans & ~(ans-1)


```
int[] noSWhichOccursOnce(int[] A)
{
    int axorB = 0;
    for (int ele: A)
        axorB = axorB ^ ele;        // XOR of every element.

    int lastBit = axorB & ~(axorB-1);

    int xA = 0, xB = 0;
    for (int ele: A)
    {
        if (ele & lastBit != 0)
        {
            xA = xA ^ ele;
        }
```

```
        else
        {
            xB = xB ^ ek;
        }
    }
    int[] output = new int[2];
    output[0] = Math.min(xA, xB);
    output[1] = Math.max(xA, xB);
    return output;
}
```