Q Matrix Game

Problem Description

Write a program to input two integers **N and M.** Now you have a 2-D array A of size N * M. It has all the integers from 1 to N * M exactly once and they are inserted in this 2-D array sequentially in a row major order.

E.g. Suppose N = 2 and M = 3, then 2-D array is {{1,2,3},{4,5,6}}

Now you have Q queries of following four types:

- 1 C1 C2: Swap elements of Column C1 with Column C2.
- 2 R1 R2: Swap elements of Row R1 with Row R2.
- 3 X1 Y1 X2 Y2: Print the Bitwise OR of element A[X1][Y1] with A[X2][Y2] in updated 2-D array.
- 4 X1 Y1 X2 Y2: Print the Bitwise AND of element A[X1][Y1] with A[X2][Y2] in updated 2-D array.

N, M. , A = N * M.

N = 2 , M = 3.

	C1	C2	C3
R1	1	2	3
R2	4	5	6

1 to N * M

1 to 6

	C1	C2	C3
R1	2	1	3
R2	5	4	6

	C1	C2	C3
R1	5	4	6
R2	2	1	3

Q queries.

(1) → 1 1 2 → swap C1 with C2

(2) → 2 1 2 → swap R1 with R2

(3) → 3 1 2 3 2 → A[1][2] (OR) A[3][2].

(4) → 4 2 2 3 2 → A[4][2] (AND) A[3][2].

Brute force :-

Simply, do whatever the question is asking for.

T.C = Q queries. → O(n) + O(m) → O(1), Q3, Q4 → O(1).

O(n): swapping columns. 1

O(m): swapping rows. 2.

O(1)

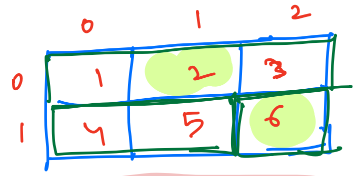
O(Q) + O(m) + O(n) → Q (m + n)

T.C = O(Q(m+n)) + mn. → for creating matrix.

S.C = O(M * N)

Let's optimise.

$N=2, M=3$



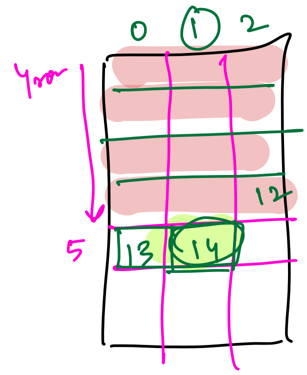
$A[1][2] = i*m + j + 1$

$= 1*3 + 2 + 1 = 3+3 = 6$

$A[1][1] = 3*1 + 1 + 1 = 5$

$A[0][1] = 3*0 + 1 + 1 = 2.$

$$\boxed{A[i][j] = i*m + j + 1}$$



$4*3 + j + 1$

$4*3 + 1 + 1$

$12+2$

$= 14$

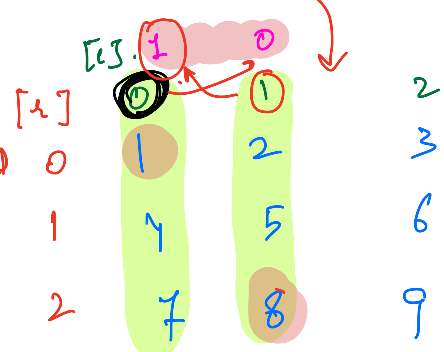
N → rows

M → columns.

Auxilliary arrays.

r[] = | 0 | 1 | 2 |

c[] = | 0 | 1 | 2 |



1 → swap 1 and 2.

Row 1 and col 1.

$A[0][0] = 1$

After swapping.

$A[0][0] = 2.$

c[] = {0, 1, 2}

c[] = {1, 0, 2}.

$A[0][0] = r[0]*m + c[0] + 1$

$= 0*m + 0 + 1 = 1$

$A[0][0] = 0*m + 1 + 1 = 2$

$A[2,1] = 2*3 + 1 + 1$

$= 8$

$A[2,1] = 2*3 + 0 + 1 = 7$

$O(m*n) \to$ ✗

$O(m+n)$ ✓

T.C =

Q. Generate Array

Problem Description

You are given an array **A** of size N and you are required to generate another array **B** of size N.

You have to find minimum value of B[i] for which Summation of (A[i] & B[i]) for i = 0 to i = N-1 is minimum possible. Also, you have to select B[i] such that (A[i] & B[i]) is a positive value.

A = [1, 2, 3]
B = [B_0, B_1, B_2]

N = 3.

Bitwise AND.

$\sum_{i=0}^{N-1} (A[i] \,\&\, B[i]) \rightarrow$ minimise this.

Select B[i].
A[i] & B[i] is +ve.

Can B[i] be 0 ? → Never

① A[i], B[i] > 0
② A[i] & B[i] → minimum
③ Choose least B[i]

A = [1, 2, 3]
B = [B_0, B_1, B_2] [1, 2, 1].

A = 0 0 1
B = 0 0 1 = 1
0 0 1 = 1

A = 0 1 0 = 2
B = 0 1 0 = 2
0 1 0 = 0 → allowed? NO.

A = 0 1 1 = 3
B = 0 0 1 =
0 0 1 = 1.

Sum = 1+2+1 = 4 → minimum

A[i] = 1 0 1 1 0 0
0 0 0 1 0 0 = 4
0 0 0 1 0 0 = 4

A[i] & B[i] = 4.
↓
4.

Pseudo Code:-

```
public int[] solve ( int[] A) {
        int B[] = new int [n];

        for( i=0; i<n ; i++) {
                int num=1;
                while ( ( num & A[i]) == 0) {
                        num <<= 1;
                }

                B[i] == num;
        }

        return B;
}
```

Dry run:

A = [2, 8, 9]

A[i]	num
2	1, 2
8	1, 2, 4, 8
9	1

B[ ] = [2, 8, 1].

final = 2, 8, 1

```
 0 1 0 = A[i].
 0 1 0 = num
 -----
 0 1 0
```

```
 1 0 0 0 = A[i].
 1 0 0 0 = num
 -------
 1 0 0 0
```

```
 1 0 0 1
 0 0 0 1
 -------
 0 0 0 1 =
```

Q. Little Pony and Maximum Element → Very easy

Problem Description

Little Ponny is given an array, **A**, of **N** integers. In a particular operation, he can set any element of the array equal to **-1**.

He wants your help in finding out the minimum number of operations required such that the maximum element of the resulting array is **B**. If it is not possible, then return **-1**.

A = [2, 4, 3, 1, 5] → N integers.

B = 3

should become greatest.

Triangle Division.
↳ uses FFT
→ skip this.

operation:- set any value to -1 in the array.

[2, 4, 3, 1, 5]

4 → -1, 5 → -1

Edge case:- If B is not present in A[]. → return -1.

Matrix