SET MATRIX ZEROES

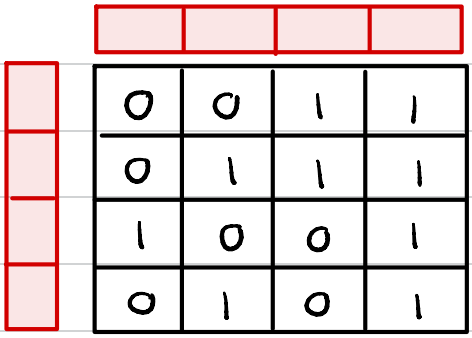
✡ In this problem, we are supposed to find 0's occuring in a 2D array and then replace all the element's occuring in that 0's row and column with 0. But this is only applicable to 0's that existed in the main array.

In the brute force solution, we iterate through the entire matrix, whenever we find a 0, we flip all the non 0 elements in its row & column to -1 (to avoid adding unwanted zeroes). Then at the end we make another iteration through the matrix and change all -1 to 0.

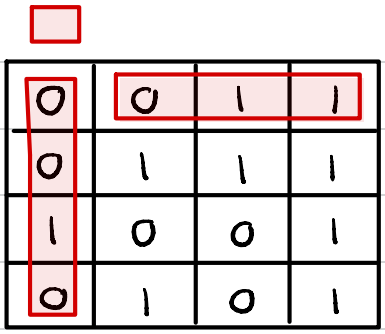
Better Solution is to keep track of which row and which column are to be turned into 0's by storing them in an array and making it happen in the next iteration. Time complexity is 2mn for a matrix of order m × n.

Time complexity can barely be optimized as traversing through a 2D matrix is always of the order N^2. What we do is we change the arrays that

were tracking if a row|column is to be changed and using them inside the matrix.



will become



The elements inside the matrix will keep track of whichever row|column is to be changed and then we will reverse iterate through the matrix to change the 1's back to 0's.

PseudoCode :

```
zeroMatrix(arr, n, m) {
    int col0 = arr[0][0]
    for(int i = 0 → n) {
        for(int j = 0 → m) {
            if(arr[i][j] == 0) {
                arr[i][0] = 0
                if(j == 0) {
                    col0 = 0
                } else {
                    arr[0][j] = 0
                }
            }
        }
    }
}
```

```
for (i = 1 → n) {
    for (j = 1 → m) {
        if (arr[i][j] != 0) {
            if (arr[0][j] == 0 or arr[i][0] == 0)
                arr[i][j] = 0
        }
    }
}
if (arr[0][0] == 0) {
    for (i = 0 → m) {
        arr[0][i] = 0
    }
}
if (col0 == 0) {
    for (i = 0 → n) {
        arr[i][0] = 0
    }
}
return arr
}
```