A Boolean Matrix Question

Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is 1 (or true) then make all the cells of ith row and jth column as 1.

```
Example 1
The matrix
1 0
0 0
should be changed to following
1 1
1 0

Example 2
The matrix
0 0 0
0 0 1
should be changed to following
0 0 1
1 1 1

Example 3
The matrix
1 0 0 1
0 0 1 0
0 0 0 0
should be changed to following
1 1 1 1
1 1 1 1
1 0 1 1
```

We strongly recommend that you click here and practice it, before moving on to the solution.

Method 1 (Use two temporary arrays)

1) Create two temporary arrays row[M] and col[N]. Initialize all values of row[] and col[] as 0.

2) Traverse the input matrix mat[M][N]. If you see an entry mat[i][j] as true, then mark row[i] and col[j] as true.

3) Traverse the input matrix mat[M][N] again. For each entry mat[i][j], check the values of row[i] and col[j]. If any of the two values (row[i] or col[j]) is true, then mark mat[i][j] as true.

Thanks to Dixit Sethi for suggesting this method.

```c
#include <stdio.h>
#define R 3
#define C 4

void modifyMatrix(bool mat[R][C])
{
    bool row[R];
    bool col[C];

    int i, j;


    /* Initialize all values of row[] as 0 */
    for (i = 0; i < R; i++)
    {
       row[i] = 0;
    }


    /* Initialize all values of col[] as 0 */
    for (i = 0; i < C; i++)
    {
       col[i] = 0;
    }


    /* Store the rows and columns to be marked as 1 in row[] and col[]
       arrays respectively */
    for (i = 0; i < R; i++)
    {
        for (j = 0; j < C; j++)
        {
            if (mat[i][j] == 1)
            {
                row[i] = 1;
                col[j] = 1;
            }
        }
    }

    /* Modify the input matrix mat[] using the above constructed row[] and
       col[] arrays */
    for (i = 0; i < R; i++)
    {
        for (j = 0; j < C; j++)
        {
            if ( row[i] == 1 || col[j] == 1 )
            {
                mat[i][j] = 1;
            }
        }
    }
}

/* A utility function to print a 2D matrix */
void printMatrix(bool mat[R][C])
{
    int i, j;
    for (i = 0; i < R; i++)
    {
        for (j = 0; j < C; j++)
        {
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }
}
```

```c
/* Driver program to test above functions */
int main()
{
    bool mat[R][C] = { {1, 0, 0, 1},
        {0, 0, 1, 0},
        {0, 0, 0, 0},
    };

    printf("Input Matrix \n");
    printMatrix(mat);

    modifyMatrix(mat);

    printf("Matrix after modification \n");
    printMatrix(mat);

    return 0;
}
```

Run on IDE

Output:

```
Input Matrix
1 0 0 1
0 0 1 0
0 0 0 0
Matrix after modification
1 1 1 1
1 1 1 1
1 0 1 1
```

Time Complexity: O(M*N)

Auxiliary Space: O(M + N)

Method 2 (A Space Optimized Version of Method 1)

This method is a space optimized version of above method 1. This method uses the first row and first column of the input matrix in place of the auxiliary arrays row[] and col[] of method 1. So what we do is: first take care of first row and column and store the info about these two in two flag variables rowFlag and colFlag. Once we have this info, we can use first row and first column as auxiliary arrays and apply method 1 for submatrix (matrix excluding first row and first column) of size (M-1)*(N-1).

1) Scan the first row and set a variable rowFlag to indicate whether we need to set all 1s in first row or not.

2) Scan the first column and set a variable colFlag to indicate whether we need to set all 1s in first column or not.

3) Use first row and first column as the auxiliary arrays row[] and col[] respectively, consider the matrix as submatrix starting from second row and second column and apply method 1.

4) Finally, using rowFlag and colFlag, update first row and first column if needed.

Time Complexity: O(M*N)

Auxiliary Space: O(1)

Thanks to Sidh for suggesting this method.