Common elements in all rows of a given matrix

Given an m x n matrix, find all common elements present in all rows in O(mn) time and one traversal of matrix.

Example:

```
Input:
mat[4][5] = {{1, 2, 1, 4, 8},
             {3, 7, 8, 5, 1},
             {8, 7, 7, 3, 1},
             {8, 1, 2, 7, 9},
            };

Output:
1 8 or 8 1
8 and 1 are present in all rows.
```

We strongly recommend you to minimize your browser and try this yourself first.

A better solution is to sort all rows in the matrix and use similar approach as discussed here. Sorting will take O(mnlogn) time and finding common elements will take O(mn) time. So overall time complexity of this solution is O(mnlogn)

Can we do better?

The idea is to use maps. We initially insert all elements of the first row in an map. For every other element in remaining rows, we check if it is present in the map. If it is present in the map and is not duplicated in current row, we increment count of the element in map by 1, else we ignore the element. If the currently traversed row is the last row, we print the element if it has appeared m-1 times before.

Below is C++ implementation of the idea.

```
// A Program to prints common element in all
// rows of matrix
#include <iostream>
#include <unordered_map>
using namespace std;

// Specify number of rows and columns
#define M 4
#define N 5

// prints common element in all rows of matrix
void printCommonElements(int mat[M][N])
{
    unordered_map<int, int> mp;

    // initalize 1st row elements with value 1
    for (int j = 0; j < N; j++)
        mp[mat[0][j]] = 1;

    // traverse the matrix
    for (int i = 1; i < M; i++)
    {
        for (int j = 0; j < N; j++)
        {
            // If element is present in the map and
            // is not duplicated in current row.
            if (mp[mat[i][j]] == i)
            {
               // we increment count of the element
               // in map by 1
                mp[mat[i][j]] = i + 1;

                // If this is last row
                if (i==M-1)
                  cout << mat[i][j] << " ";
            }
        }
    }
}

// driver program to test above function
int main()
{
    int mat[M][N] =
    {
        {1, 2, 1, 4, 8},
        {3, 7, 8, 5, 1},
        {8, 7, 7, 3, 1},
        {8, 1, 2, 7, 9},
    };

    printCommonElements(mat);

    return 0;
}
```

Output:

```
8 1
```

The time complexity of this solution is O(m * n) and we are doing only one traversal of the matrix.