Reshape the Matrix

In MATLAB, there is a very useful function called 'reshape', which can reshape a matrix into a new one with different size but keep its original data.

You're given a matrix represented by a two-dimensional array, and two **positive** integers **r** and **c** representing the **row** number and **column** number of the wanted reshaped matrix, respectively.

The reshaped matrix need to be filled with all the elements of the original matrix in the same **row-traversing** order as they were.

If the 'reshape' operation with given parameters is possible and legal, output the new reshaped matrix; Otherwise, output the original matrix.

Example 1:

```
Input:
nums =
[[1,2],
 [3,4]]
r = 1, c = 4
Output:
[[1,2,3,4]]
Explanation:
The row-traversing of nums is [1,2,3,4]. The new reshaped matrix is a 1 * 4 matrix, f
ill it row by row by using the previous list.
```

Example 2:

```
Input:
nums =
[[1,2],
 [3,4]]
r = 2, c = 4
Output:
[[1,2],
 [3,4]]
Explanation:
There is no way to reshape a 2 * 2 matrix to a 2 * 4 matrix. So output the original m
atrix.
```

Note:

1. The height and width of the given matrix is in range [1, 100].
2. The given r and c are all positive.

Solution 1

```
public int[][] matrixReshape(int[][] nums, int r, int c) {
    int n = nums.length, m = nums[0].length;
    if (r*c != n*m) return nums;
    int[][] res = new int[r][c];
    for (int i=0;i<r*c;i++)
        res[i/c][i%c] = nums[i/m][i%m];
    return res;
}
```

written by compton_scatter original link here

Solution 2

We can use `matrix[index / width][index % width]` for both the input and the output matrix.

```
public int[][] matrixReshape(int[][] nums, int r, int c) {
    int m = nums.length, n = nums[0].length;
    if (r * c != m * n)
        return nums;
    int[][] reshaped = new int[r][c];
    for (int i = 0; i < r * c; i++)
        reshaped[i/c][i%c] = nums[i/n][i%n];
    return reshaped;
}
```

written by StefanPochmann original link here

Solution 3

```
class Solution {
public:
    vector<vector<int>> matrixReshape(vector<vector<int>>& nums, int r, int c) {
        int m = nums.size(), n = nums[0].size(), o = m * n;
        if (r * c != o) return nums;
        vector<vector<int>> res(r, vector<int>(c, 0));
        for (int i = 0; i < o; i++) res[i / c][i % c] = nums[i / n][i % n];
        return res;
    }
};
```

```
class Solution {
public:
    vector<vector<int>> matrixReshape(vector<vector<int>>& nums, int r, int c) {
        int m = nums.size(), n = nums[0].size();
        if (m * n != r * c) {
            return nums;
        }

        vector<vector<int>> res(r, vector<int>(c, 0));
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                int k = i * n + j;
                res[k / c][k % c] = nums[i][j];
            }
        }

        return res;
    }
};
```

written by alexander original link here