566 Reshape the Matrix

2018年4月15日 18:38

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, fill 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 matrix.

Note:

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

来自 <https://leetcode.com/problems/reshape-the-matrix/description/>

在MATLAB中，有一个非常有用的函数 reshape，它可以将一个矩阵重塑为另一个大小不同的新矩阵，但保留其原始数据。

给出一个由二维数组表示的矩阵，以及两个正整数r和c，分别表示想要的重构的矩阵的行数和列数。

重构后的矩阵需要将原始矩阵的所有元素以相同的**行遍历顺序**填充。

如果具有给定参数的reshape操作是可行且合理的，则输出新的重塑矩阵；否则，输出原始矩阵。

注意：

1. 给定矩阵的宽和高范围在 [1, 100]。
2. 给定的 r 和 c 都是正数。

Solution for Python3:

```python
class Solution1:
    def matrixReshape(self, nums, r, c):
        """
        :type nums: List[List[int]]
        :type r: int
        :type c: int
        :rtype: List[List[int]]
        """
        m, n = len(nums), len(nums[0])
        if m * n != r * c:
            return nums
        res = [[0]*c for row in range(r)]
        for i in range(r * c):
            res[i//c][i%c] = nums[i//n][i%n]
        return res


class Solution2:
    def matrixReshape(self, nums, r, c):
        """
        :type nums: List[List[int]]
        :type r: int
        :type c: int
        :rtype: List[List[int]]
        """
        if len(nums) * len(nums[0]) != r * c:
            return nums
        res = []
        for num in nums:
            res.extend(num)
        return [res[i:i+c] for i in range(0, r*c, c)]

class Solution3:
    def matrixReshape(self, nums, r, c):
        """
        :type nums: List[List[int]]
        :type r: int
        :type c: int
        :rtype: List[List[int]]
        """
        import numpy as np
        try:
            return np.reshape(nums, (r, c)).tolist()
        except:
            return nums

class Solution4:
    def matrixReshape(self, nums, r, c):
        """
        :type nums: List[List[int]]
        :type r: int
        :type c: int
        :rtype: List[List[int]]
        """
        if r * c != len(nums) * len(nums[0]):
            return nums
        it = itertools.chain(*nums)
        return [list(itertools.islice(it, c)) for _ in range(r)]
```

Solution for C++:

```cpp
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 < r * c; i++) {
            res[i/c][i%c] = nums[i/n][i%n];
        }
        return res;
    }
};
```

Appendix:

Python 初始化二维列表：

1) [[0] * n] * m 这是错误的，因为[0]*n是一个一位数组对象，*m只是把对象引用复制了m次,即实际这m个值都指向同一个对象。之后不论怎么修改某一行的某个值，改行所有值都会变成一样。
2) [[0] * n for x in range(m)]:这样做二维矩阵列表中的每个值都指向不同对象。