661 Image Smoother

2018年4月22日 14:22

Given a 2D integer matrix M representing the gray scale of an image, you need to design a smoother to make the gray scale of each cell becomes the average gray scale (rounding down) of all the 8 surrounding cells and itself. If a cell has less than 8 surrounding cells, then use as many as you can.

Example 1:

Input:

[[1,1,1],
 [1,0,1],
 [1,1,1]]

Output:

[[0, 0, 0],
 [0, 0, 0],
 [0, 0, 0]]

Explanation:

For the point (0,0), (0,2), (2,0), (2,2): floor(3/4) = floor(0.75) = 0
For the point (0,1), (1,0), (1,2), (2,1): floor(5/6) = floor(0.83333333) = 0
For the point (1,1): floor(8/9) = floor(0.88888889) = 0

Note:

1. The value in the given matrix is in the range of [0, 255].
2. The length and width of the given matrix are in the range of [1, 150].

来自 <https://leetcode.com/problems/image-smoother/description/>

包含整数的二维矩阵 M 表示一个图片的灰度。你需要设计一个平滑器来让每一个单元的灰度成为平均灰度 (向下舍入)，平均灰度的计算是周围的8个单元和它本身的值求平均，如果周围的单元格不足八个，则尽可能多的利用它们。

注意:

1. 给定矩阵中的整数范围为 [0, 255]。
2. 矩阵的长和宽的范围均为 [1, 150]。

Solution for Python3:

```python
class Solution:
    def imageSmoother(self, M):
        """
        :type M: List[List[int]]
        :rtype: List[List[int]]
        """
        R = len(M)
        C = len(M[0]) if R else 0
        res = copy.deepcopy(M)
        for x in range(R):
           for y in range(C):
               neighbors = [
               M[i][j]
               for i in (x-1, x, x+1)
               for j in (y-1, y, y+1)
               if 0 <= i < R and 0 <= j < C]
               res[x][y] = sum(neighbors) // len(neighbors)
        return res
```

Solution for C++:

```cpp
class Solution1 {
public:
    vector<vector<int>> imageSmoother(vector<vector<int>>& M) {
        int R = M.size(), C = M[0].size();
        vector<vector<int>> res(R, vector<int>(C, 0));
        for (int i = 0; i < R; i++) {
```

```
            for (int j = 0; j < C; j++) {
                int cnt = 0;
                for (int nr = i - 1; nr <= i + 1; nr++) {
                    for (int nc = j - 1; nc <= j + 1; nc++) {
                        if (nr >= 0 && nr < R && 0 <= nc && nc < C) {
                            res[i][j] += M[nr][nc];
                            cnt++;
                        }
                    }
                }
                res[i][j] /= cnt;
            }
        }
        return res;
    }
};

// 把每个位置计算的结果存放在高位，而低8位存放该位置原来的值
class Solution2 {
public:
    vector<vector<int>> imageSmoother(vector<vector<int>>& M) {
        int R = M.size(), C = M[0].size();
        if (R == 0 || C == 0)
            return {{}};
        vector<vector<int>> dirs =
{{0,1},{0,-1},{1,0},{-1,0},{-1,-1},{1,1},{-1,1},{1,-1}};
        for (int i = 0; i < R; i++) {
            for (int j = 0; j < C; j++) {
                int sum = M[i][j], cnt = 1;
                for (int k = 0; k < dirs.size(); k++) {
                    int x = i + dirs[k][0], y = j + dirs[k][1];
                    if (x < 0 || x >= R || y < 0 || y >= C)
                        continue;
                    sum += (M[x][y] & 0xFF);
                    cnt++;
                }
                M[i][j] |= ((sum / cnt) << 8);
            }
        }
        for (int i = 0; i < R; i++) {
            for (int j = 0; j < C; j++) {
                M[i][j] >>= 8;
            }
        }
        return M;
    }
};
```