

```

1  /*
2  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。
3  示例 1:
4  输入:
5  [[1,1,1],
6   [1,0,1],
7   [1,1,1]]
8  输出:
9  [[0, 0, 0],
10 [0, 0, 0],
11 [0, 0, 0]]
12 解释:
13 对于点 (0,0), (0,2), (2,0), (2,2): 平均(3/4) = 平均(0.75) = 0
14 对于点 (0,1), (1,0), (1,2), (2,1): 平均(5/6) = 平均(0.83333333) = 0
15 对于点 (1,1): 平均(8/9) = 平均(0.88888889) = 0
16 注意:
17     给定矩阵中的整数范围为 [0, 255]。
18     矩阵的长和宽的范围均为 [1, 150]。
19 来源: 力扣 (LeetCode)
20 链接: https://leetcode-cn.com/problems/image-smoother
21 著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
22 */

```

分析:

- 方法一:非原地解法:通过检测下标索引是否越界来计算总和和均值.
- 方法二:int的原地解法
 - 由于M的数据的范围是0 ~ 255,只用到int四个字节中的一个字节.
 - 我们可以从其他空闲字节中选取一个作为计算结果的存储空间,计算完毕后,把计算结果赋值给M即可.
 - 若传入的M是 `vector < vector < unsignedchar >>`,该解法失效.
 - 与操作取低八位时默认了CPU是小端.

方法一:C++_非原地解法

```

1  class solution
2  {
3
4      private:
5          int __neighborAver(vector<vector<int>>& M , int rows , int cols ,
6          int row , int col)
7          {
8              int temp    =  M[row][col];
9              int count   =  1;
10
11              /*上一行*/
12              if(row-1>=0)
13              {
14                  temp += M[row-1][col];
15                  count++;
16              }
17
18              /*下一行*/
19              if(row+1<rows)
20              {
21                  temp += M[row+1][col];
22                  count++;
23              }
24
25              /*左一列*/
26              if(col-1>=0)
27              {
28                  temp += M[row][col-1];
29                  count++;
30              }
31
32              /*右一列*/
33              if(col+1<cols)
34              {
35                  temp += M[row][col+1];
36                  count++;
37              }
38
39              return temp/count;
40          }
41
42      public:
43          vector<vector<int>> smoothImage(vector<vector<int>> M) {
44              int rows = M.size();
45              int cols = M[0].size();
46              vector<vector<int>> ans(rows, vector<int>(cols));
47              for(int i = 0; i < rows; i++)
48                  for(int j = 0; j < cols; j++)
49                      ans[i][j] = __neighborAver(M, rows, cols, i, j);
50              return ans;
51          }
52
53      };

```

```

15         if(col-1 >=0)
16         {
17             temp += M[row-1][col-1];
18             count++;
19         }
20
21         if(col+1 < cols)
22         {
23             temp += M[row-1][col+1];
24             count++;
25         }
26     }
27
28     /*下一行*/
29     if(row+1< rows)
30     {
31         temp += M[row+1][col];
32         count++;
33         if(col-1 >=0)
34         {
35             temp += M[row+1][col-1];
36             count++;
37         }
38
39         if(col+1 < cols)
40         {
41             temp += M[row+1][col+1];
42             count++;
43         }
44     }
45
46     /*中间一行*/
47     if(col-1 >=0)
48     {
49         temp += M[row][col-1];
50         count++;
51     }
52
53     if(col+1 < cols)
54     {
55         temp += M[row][col+1];
56         count++;
57     }
58
59     return (int)(temp/count);
60 }
61
62 public:
63     vector<vector<int>> imageSmoother(vector<vector<int>>& M)
64     {
65         int r    = M.size();
66         int c    = M[0].size();
67
68         vector<vector<int>> vvi(r,vector<int>(c,0));
69
70         for(int i = 0; i < r; i++)
71         {
72             for(int j = 0 ; j < c; j++)

```

```

73         {
74             vvi[i][j] = __neighborAver(M,r,c,i,j);
75         }
76     }
77     return vvi;
78 }
79 };
80
81
82
83 /*
84 执行结果:
85 通过
86 显示详情
87 执行用时 :168 ms, 在所有 cpp 提交中击败了97.11% 的用户
88 内存消耗 :17.4 MB, 在所有 cpp 提交中击败了88.18%的用户
89 */

```

方法二:C++_原地解法

```

1  class Solution
2  {
3
4      private:
5          void __neighborAver(vector<vector<int>>& M , int rows , int cols ,
6              int row , int col)
7          {
8              int temp    =  (M[row][col] & 0xff);
9              int count   =  1;
10
11              /*上一行*/
12              if(row-1>=0)
13              {
14                  temp += (M[row-1][col] & 0xff);
15                  count++;
16                  if(col-1 >=0)
17                  {
18                      temp += (M[row-1][col-1] & 0xff);
19                      count++;
20                  }
21
22                  if(col+1 < cols)
23                  {
24                      temp += (M[row-1][col+1] & 0xff);
25                      count++;
26                  }
27              }
28
29              /*下一行*/
30              if(row+1< rows)
31              {
32                  temp += (M[row+1][col] & 0xff);
33                  count++;
34                  if(col-1 >=0)
35                  {
36                      temp += (M[row+1][col-1] & 0xff);

```

```

36         count++;
37     }
38
39     if(col+1 < cols)
40     {
41         temp += (M[row+1][col+1]& 0xff);
42         count++;
43     }
44 }
45
46 /*中间一行*/
47 if(col-1 >=0)
48 {
49     temp += (M[row][col-1]& 0xff);
50     count++;
51 }
52
53 if(col+1 < cols)
54 {
55     temp += (M[row][col+1] & 0xff);
56     count++;
57 }
58
59 (( unsigned char*)&M[row][col]))[1] = temp/count;
60 // //M[i][j] |= (temp/count)<<8;
61 }
62
63 public:
64     vector<vector<int>> imagesmoother(vector<vector<int>>& M)
65     {
66         int r    = M.size()    ;
67         int c    = M[0].size() ;
68         int i    = 0           ;
69         int j    = 0           ;
70
71         /*计算8~15bit*/
72         for(i = 0; i < r; i++)
73         {
74             for(j = 0 ; j < c; j++)
75             {
76                 __neighborAver(M,r,c,i,j);
77             }
78         }
79
80         /*保留8~15bit--->0~7bit*/
81         for(i = 0; i < r; i++)
82         {
83             for(j = 0 ; j < c; j++)
84             {
85                 //M[i][j] >>= 8;
86                 M[i][j] = (( unsigned char*)&M[i][j]))[1];
87             }
88         }
89
90
91         return M;
92     }
93 };

```

```
94
95  /*
96  执行结果:
97  通过
98  显示详情
99  执行用时 :264 ms, 在所有 cpp 提交中击败了31.23% 的用户
100  内存消耗 :17.6 MB, 在所有 cpp 提交中击败了79.09%的用户
101  https://leetcode-cn.com/submissions/detail/37351245/
102  */
```

AlimyBreak
2019.11.22