```
/*
给定一个 m x n 的矩阵，如果一个元素为 0，则将其所在行和列的所有元素都设为 0。请使用原地算法。

示例 1:

输入:
[
  [1,1,1],
  [1,0,1],
  [1,1,1]
]
输出:
[
  [1,0,1],
  [0,0,0],
  [1,0,1]
]
示例 2:
输入:
[
  [0,1,2,0],
  [3,4,5,2],
  [1,3,1,5]
]
输出:
[
  [0,0,0,0],
  [0,4,5,0],
  [0,3,1,0]
]
进阶:
    一个直接的解决方案是使用  O(mn) 的额外空间，但这并不是一个好的解决方案。
    一个简单的改进方案是使用 O(m + n) 的额外空间，但这仍然不是最好的解决方案。
    你能想出一个常数空间的解决方案吗？

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/set-matrix-zeroes
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。
*/
```

分析:

- 方法一:遍历矩阵,使用vector<pair<int,int>>保存出现0的位置,然后再遍历$vpii$,将对应行列全部置0,空间复杂度$O(mn)$;
- 方法二:遍历矩阵,使用两个map<int,int>来保存出现过0的行和列,然后再遍历两个mii,将对应行列全部置0,空间复杂度$O(mn)$.
- 方法三:遍历矩阵做标记,这个算法有缺陷，数组中的的数不能出现-10000和-20000,空间复杂度$O(1)$.

方法一:C++_vpii

```
class Solution
{

    private:

        void setRowZero(vector<vector<int>>& matrix, int row_number, int cols )
        {
            int i = 0 ;

            for( i = 0 ; i < cols ; i++)
            {
                matrix[row_number][i] = 0;
            }
        }


        void setColZero(vector<vector<int>>& matrix, int col_number, int rows)
        {
            int i = 0;
            for(i = 0; i < rows;i++)
            {
                matrix[i][col_number] = 0;
            }
        }


    public:
        void setZeroes(vector<vector<int>>& matrix)
        {
            int m = matrix.size();
            int n = matrix[0].size();
            vector<pair<int,int>>  vpii;
            int i = 0;
            int j = 0;

            /*1.遍历找0的位置*/
            for(i = 0 ; i < m ; i++)
            {
                for(j = 0 ; j < n ; j ++)
                {
                    if(matrix[i][j]==0)
                    {
                        vpii.push_back(make_pair(i,j));
                    }
                }
            }

            /*2.置零*/
            for(i = 0; i < vpii.size();i++)
            {
                setRowZero(matrix,vpii[i].first,n);
                setColZero(matrix,vpii[i].second,m);
            }
```

```
        }

};


/*
执行结果:
通过
显示详情
执行用时 :60 ms, 在所有 cpp 提交中击败了74.42% 的用户
内存消耗 :11.5 MB, 在所有 cpp 提交中击败了30.16%的用户
*/
```

方法二:C++_mii

```
class Solution
{

    private:

        void setRowZero(vector<vector<int>>& matrix, int row_number, int cols )
        {
            int i = 0 ;

            for( i = 0 ; i < cols ; i++)
            {
                matrix[row_number][i] = 0;
            }
        }


        void setColZero(vector<vector<int>>& matrix, int col_number, int rows)
        {
            int i = 0;
            for(i = 0; i < rows;i++)
            {
                matrix[i][col_number] = 0;
            }
        }


    public:
        void setZeroes(vector<vector<int>>& matrix)
        {
            int m = matrix.size();
            int n = matrix[0].size();
            //vector<pair<int,int>>  vpii;

            map<int,int> mii_row;
            map<int,int> mii_col;

            int i = 0;
            int j = 0;

```

```
            /*1.遍历找0的位置*/
            for(i = 0 ; i < m ; i++)
            {
                for(j = 0 ; j < n ; j ++)
                {
                    if(matrix[i][j]==0)
                    {
                        mii_row[i] = 0;
                        mii_col[j] = 0;

                    }
                }
            }

            /*2.置零*/
            map<int, int>::iterator  iter = mii_row.begin();
            while(iter!=mii_row.end())
            {
                setRowZero(matrix,iter->first,n);
                iter++;
            }

            iter = mii_col.begin();
            while(iter != mii_col.end())
            {
                setColZero(matrix,iter->first,m);
                iter++;
            }
        }
};


/*
执行结果:
通过
显示详情
执行用时 :56 ms, 在所有 cpp 提交中击败了89.67% 的用户
内存消耗 :11.6 MB, 在所有 cpp 提交中击败了11.11%的用户
*/
```

方法三:C++_标记法(有缺陷)

```
class Solution
{
    private:
        void setRowNeg(vector<vector<int>>& matrix, int row_number, int cols )
        {
            int i = 0 ;
            for( i = 0 ; i < cols ; i++)
            {
                if(matrix[row_number][i]==0 || matrix[row_number][i]==-20000)
                {
                    matrix[row_number][i] = -20000;
                }
```

```
                else
                {
                    matrix[row_number][i] = -10000;
                }
            }
        }

        void setColNeg(vector<vector<int>>& matrix, int col_number, int
rows)
        {
            int i = 0;
            for(i = 0; i < rows;i++)
            {
                if(matrix[i][col_number]==0 || matrix[i]
[col_number]==-20000)
                {
                    matrix[i][col_number] = -20000;
                }
                else
                {
                    matrix[i][col_number] = -10000;
                }
            }
        }

    public:
        void setZeroes(vector<vector<int>>& matrix)
        {
            int m   = matrix.size()     ;
            int n   = matrix[0].size()  ;
            int i   = 0                 ;
            int j   = 0                 ;

            /*1.遍历找0的位置,并置负数*/
            for(i = 0 ; i < m ; i++)
            {
                for(j = 0 ; j < n ; j++)
                {
                    if(matrix[i][j]==0 || matrix[i][j]==-20000 )
                    {
                        setRowNeg(matrix,i,n);
                        setColNeg(matrix,j,m);
                    }
                }
            }

            /*2.挑出负数置0*/
            for(i = 0 ; i < m ; i++)
            {
                for(j = 0 ; j < n ; j++)
                {
                    if(matrix[i][j]==-10000 || matrix[i][j]==-20000)
                    {
                        matrix[i][j] = 0;
                    }
                }
            }
        }
```

```
};
/*
执行结果:
通过
显示详情
执行用时 :96 ms, 在所有 cpp 提交中击败了26.06% 的用户
内存消耗 :11.4 MB, 在所有 cpp 提交中击败了66.83%的用户
*/
```

AlimyBreak
2019.11.19