# 题目

给你一个 m x n 大小的矩阵 grid ，由若干正整数组成。

1、执行下述操作，直到 grid 变为空矩阵：

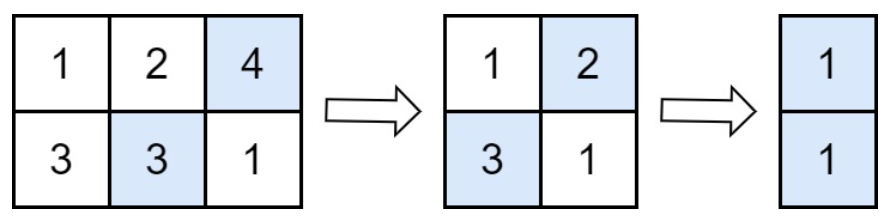
2、从每一行删除值最大的元素。如果存在多个这样的值，删除其中任何一个。

将删除元素中的最大值与答案相加。

注意：每执行一次操作，矩阵中列的数据就会减1。

返回执行上述操作后的答案。

示例 1：



输入：grid = [[1,2,4],[3,3,1]]

输出：8

解释：上图展示在每一步中需要移除的值。

- 在第一步操作中，从第一行删除4，从第二行删除3（注意，有两个单元格中的值为3，我们可以删除任一）。在答案上加4。

- 在第二步操作中，从第一行删除2，从第二行删除3。在答案上加3。

- 在第三步操作中，从第一行删除1，从第二行删除1。在答案上加1。

最终，答案= 4 + 3 + 1 = 8。

示例 2：

输入：grid = [[10]]

输出：10

解释：上图展示在每一步中需要移除的值。

- 在第一步操作中，从第一行删除10。在答案上加10。

最终，答案= 10。

提示：

m == grid.length

n == grid[i].length

1 <= m, n <= 50

1 <= grid[i][j] <= 100

# 分析

## 方法一：模拟

## 方法二：排序（推荐）

**思路：**

**1、操作描述：**

对矩阵 grid 中的每一行，找到该行的最大值并删除（如果有多个最大值，随便删除一个）。

找到所有被删除元素中的最大值，将其加入答案中。

重复上述步骤，直到矩阵为空。

**2、关键点：**

对于矩阵中的每一行，始终找出当前行的最大值。

每列中每次选取最大的值要符合操作顺序。

**3、优化思路：**

可以先对每一行进行降序排序，方便后续快速取出每行的最大值。

从列的角度取每一轮的最大值，避免重复扫描。

**代码：**

class Solution {

public:

int deleteGreatestValue(vector<vector<int>>& grid) {

int m = grid.size(), n = grid[0].size();

// 对每一行进行排序（降序）

for (auto& row : grid) {

sort(row.begin(), row.end(), greater<int>());

}

int result = 0;

// 按列计算每一列的最大值

for (int col = 0; col < n; ++col) {

int maxInColumn = 0;

for (int row = 0; row < m; ++row) {

maxInColumn = max(maxInColumn, grid[row][col]);

}

result += maxInColumn;

}

return result;

}

};

**或：**

class Solution {

public:

int deleteGreatestValue(vector<vector<int>>& grid) {

// 对每一行进行排序（降序）

for (auto& row : grid) {

sort(row.rbegin(), row.rend());

}

int result = 0;

// 遍历列

for (int col = 0; col < grid[0].size(); ++col) {

int maxInColumn = 0;

// 找到每一列的最大值

for (int row = 0; row < grid.size(); ++row) {

maxInColumn = max(maxInColumn, grid[row][col]);

}

// 累加当前列的最大值

result += maxInColumn;

}

return result;

}

};

另一种写法：

class Solution {

public:

int deleteGreatestValue(vector<vector<int>>& grid) {

int m = grid.size(), n = grid[0].size();

for (int i = 0; i < m; i++) {

sort(grid[i].begin(), grid[i].end());

}

int res = 0;

// 遍历列

for (int j = 0; j < n; j++) {

int mx = 0;

// 遍历行

for (int i = 0; i < m; i++) {

mx = max(mx, grid[i][j]);

}

res += mx;

}

return res;

}

};