# 题目

给你一个下标从 0 开始的二维整数数组 nums 。一开始你的分数为 0 。你需要执行以下操作直到矩阵变为空：

1、矩阵中每一行选取最大的一个数，并删除它。如果一行中有多个最大的数，选择任意一个并删除。

2、在步骤 1 删除的所有数字中找到最大的一个数字，将它添加到你的 分数 中。

请你返回最后的 分数 。

示例 1：

输入：nums = [[7,2,1],[6,4,2],[6,5,3],[3,2,1]]

输出：15

解释：第一步操作中，我们删除 7 ，6 ，6 和 3 ，将分数增加 7 。下一步操作中，删除 2 ，4 ，5 和 2 ，将分数增加 5 。最后删除 1 ，2 ，3 和 1 ，将分数增加 3 。所以总得分为 7 + 5 + 3 = 15 。

示例 2：

输入：nums = [[1]]

输出：1

解释：我们删除 1 并将分数增加 1 ，所以返回 1 。

提示：

1 <= nums.length <= 300

1 <= nums[i].length <= 500

0 <= nums[i][j] <= 10^3

# 分析

## 方法一：模拟

## 方法二：排序

思路：

由于每次删除操作中，每行删除的元素即为当前行中的最大值，因此我们可以直接将每行的元素按照从大到小排序，然后按照列遍历矩阵，每次删除操作得分即为当前列的最大值，因此最终得分即为所有列中的最大值之和。

代码：

class Solution {

public:

int matrixSum(vector<vector<int>>& nums) {

int res = 0;

// 获取二维数组的行数和列数

int m = nums.size(); // 行数

int n = nums[0].size(); // 列数

// 对每一行进行排序

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

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

}

// 计算每一列的最大值之和

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

int maxVal = 0;

// 遍历每一行，找到当前列的最大值

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

maxVal = max(maxVal, nums[i][j]);

}

// 将最大值累加到结果中

res += maxVal;

}

return res;

}

};