# 题目

给定 m 个数组，每个数组都已经按照升序排好序了。

现在你需要从两个不同的数组中选择两个整数（每个数组选一个）并且计算它们的距离。两个整数 a 和 b 之间的距离定义为它们差的绝对值 |a-b| 。

返回最大距离。

示例 1：

输入：[[1,2,3],[4,5],[1,2,3]]

输出：4

解释：

一种得到答案 4 的方法是从第一个数组或者第三个数组中选择 1，同时从第二个数组中选择 5 。

示例 2：

输入：arrays = [[1],[1]]

输出：0

提示：

m == arrays.length

2 <= m <= 105

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

-104 <= arrays[i][j] <= 104

arrays[i] 以 升序 排序。

所有数组中最多有 105 个整数。

# 分析

要解决从多个升序数组中选择两个不同数组的整数，求最大距离（绝对值差）的问题，核心是利用数组的升序特性——每个数组的极值（最小值在开头，最大值在结尾）是产生最大距离的关键，无需遍历所有元素。

解题思路：

1、核心观察：

由于每个数组已升序排序，其最小值为数组第一个元素（arr[0]），最大值为最后一个元素（arr.back()）。任意两个不同数组的最大距离，必然是某数组的最大值与另一数组的最小值的绝对值（因为最大值和最小值的差会最大）。

2、高效计算策略：

- 遍历所有数组，记录当前已遍历数组的全局最小值（min\_val） 和全局最大值（max\_val）。

- 对每个数组，计算两种可能的最大距离：

1. 当前数组的最大值与全局最小值的差（curr\_max - min\_val）；

2. 全局最大值与当前数组的最小值的差（max\_val - curr\_min）。

- 每次遍历后，更新全局最小值和全局最大值，确保后续数组能基于最新的全局极值计算距离。

- 最终的最大距离即为所有计算结果中的最大值。

3、边界处理：

初始时，将第一个数组的最小值和最大值作为初始全局极值，从第二个数组开始遍历（避免选择同一数组的极值，确保“两个不同数组”的约束）。

代码：

class Solution {

public:

int maxDistance(vector<vector<int>>& arrays) {

int m = arrays.size();

if (m < 2) return 0; // 题目保证m>=2，可省略

// 初始化：第一个数组的最小值和最大值作为初始全局极值

int global\_min = arrays[0][0];

int global\_max = arrays[0].back();

int max\_dist = 0; // 存储最大距离

// 从第二个数组开始遍历，计算并更新最大距离

for (int i = 1; i < m; ++i) {

int curr\_min = arrays[i][0]; // 当前数组的最小值

int curr\_max = arrays[i].back();// 当前数组的最大值

// 计算两种可能的最大距离（当前数组与之前所有数组的极值差）

int dist1 = curr\_max - global\_min; // 当前最大值 - 全局最小值

int dist2 = global\_max - curr\_min; // 全局最大值 - 当前最小值

max\_dist = max(max\_dist, max(dist1, dist2));

// 更新全局极值（纳入当前数组的极值，供后续数组使用）

global\_min = min(global\_min, curr\_min);

global\_max = max(global\_max, curr\_max);

}

return max\_dist;

}

};

代码解释：

1、初始化全局极值：

从第一个数组提取global\_min（首元素）和global\_max（尾元素），作为后续数组的对比基准。

2、遍历后续数组：

对每个数组（从第二个开始），先获取其自身的curr\_min和curr\_max，然后计算该数组与之前所有数组的两种最大可能距离（当前最大值 vs 历史最小，历史最大 vs 当前最小），并更新全局最大距离max\_dist。

3、更新全局极值：

将当前数组的极值纳入全局范围，确保后续数组能与所有已遍历数组的极值对比，覆盖所有“不同数组”的组合。

复杂度分析

- 时间复杂度：O(m)，其中m是数组的个数。每个数组仅需访问首、尾两个元素（O(1) per array），遍历所有数组总耗时O(m)，无额外嵌套循环。

- 空间复杂度：O(1)，仅使用常数个变量（global\_min、global\_max、max\_dist等），不依赖输入规模。

该方法充分利用数组的升序特性，避免冗余计算，高效解决了“最大距离”问题，尤其适合题目中m达10^5的大规模输入场景。