# 题目

给定一个整数数组 asteroids，表示在同一行的小行星。数组中小行星的索引表示它们在空间中的相对位置。

对于数组中的每一个元素，其绝对值表示小行星的大小，正负表示小行星的移动方向（正表示向右移动，负表示向左移动）。每一颗小行星以相同的速度移动。

找出碰撞后剩下的所有小行星。碰撞规则：两个小行星相互碰撞，较小的小行星会爆炸。如果两颗小行星大小相同，则两颗小行星都会爆炸。两颗移动方向相同的小行星，永远不会发生碰撞。

示例 1：

输入：asteroids = [5,10,-5]

输出：[5,10]

解释：10 和 -5 碰撞后只剩下 10 。 5 和 10 永远不会发生碰撞。

示例 2：

输入：asteroids = [8,-8]

输出：[]

解释：8 和 -8 碰撞后，两者都发生爆炸。

示例 3：

输入：asteroids = [10,2,-5]

输出：[10]

解释：2 和 -5 发生碰撞后剩下 -5 。10 和 -5 发生碰撞后剩下 10 。

提示：

2 <= asteroids.length <= 104

-1000 <= asteroids[i] <= 1000

asteroids[i] != 0

# 分析

要解决“小行星碰撞”问题，核心思路是利用栈模拟碰撞过程：栈中始终维护“当前未碰撞的小行星”，通过判断栈顶元素与当前小行星的移动方向和大小，处理所有碰撞场景，最终栈中剩余元素即为碰撞后的结果。

解题思路

1、碰撞条件精准判断：

仅当“栈顶小行星向右（值为正）”且“当前小行星向左（值为负）”时，才会发生碰撞。其他情况（同向移动、栈顶向左而当前向右）均不会碰撞，直接将当前小行星入栈即可。

2、栈的核心作用：

栈存储“已稳定的小行星”——即栈内元素不会与左侧元素发生碰撞。遍历每个小行星时，若触发碰撞条件，则持续与栈顶元素处理碰撞，直到以下任一情况发生：

- 栈为空（当前小行星无碰撞对象，直接入栈）；

- 栈顶小行星向左（无碰撞可能，当前小行星入栈）；

- 当前小行星爆炸（栈顶小行星更大，不处理）；

- 两者均爆炸（栈顶弹出，当前小行星不处理）。

3、碰撞处理逻辑：

对于触发碰撞的栈顶（top）和当前小行星（curr），通过绝对值比较大小：

- 若 |curr| > |top|：栈顶小行星爆炸（弹出），继续判断新栈顶（可能存在连续碰撞）；

- 若 |curr| == |top|：两者均爆炸（弹出栈顶，标记当前小行星为爆炸状态）；

- 若 |curr| < |top|：当前小行星爆炸（标记状态，停止碰撞）。

代码：

class Solution {

public:

vector<int> asteroidCollision(vector<int>& asteroids) {

vector<int> stack; // 栈：存储未碰撞的小行星

for (int curr : asteroids) {

bool exploded = false; // 标记当前小行星是否在碰撞中爆炸

// 持续处理碰撞：栈非空 + 栈顶向右 + 当前向左（唯一碰撞场景）

while (!stack.empty() && stack.back() > 0 && curr < 0) {

int top = stack.back(); // 获取栈顶小行星

int curr\_abs = abs(curr);

int top\_abs = abs(top);

if (curr\_abs > top\_abs) {

// 栈顶小行星更小，爆炸（弹出），继续检查新栈顶

stack.pop\_back();

} else if (curr\_abs == top\_abs) {

// 两者大小相等，均爆炸：弹出栈顶，标记当前为爆炸

stack.pop\_back();

exploded = true;

break;

} else {

// 当前小行星更小，爆炸：标记后停止碰撞

exploded = true;

break;

}

}

// 若当前小行星未爆炸，入栈（无碰撞或碰撞后存活）

if (!exploded) {

stack.push\_back(curr);

}

}

return stack; // 栈中剩余元素即为碰撞后结果

}

};

复杂度分析：

- 时间复杂度：O(n)，其中n是小行星数量。每个小行星最多入栈1次、出栈1次，碰撞循环的总执行次数为O(n)（无冗余操作），整体为线性时间。

- 空间复杂度：O(n)。最坏情况下（所有小行星同向移动，无碰撞），栈需存储全部小行星，空间开销为O(n)。