# 题目

给定一个长度为 n 的整数数组 nums 和一个二维数组 queries，其中 queries[i] = [li, ri]。

对于每个查询 queries[i]：

在 nums 的下标范围 [li, ri] 内选择一个下标 子集。

将选中的每个下标对应的元素值减 1。

零数组 是指所有元素都等于 0 的数组。

如果在按顺序处理所有查询后，可以将 nums 转换为 零数组 ，则返回 true，否则返回 false。

示例 1：

输入： nums = [1,0,1], queries = [[0,2]]

输出： true

解释：

对于 i = 0：

选择下标子集 [0, 2] 并将这些下标处的值减 1。

数组将变为 [0, 0, 0]，这是一个零数组。

示例 2：

输入： nums = [4,3,2,1], queries = [[1,3],[0,2]]

输出： false

解释：

对于 i = 0：

选择下标子集 [1, 2, 3] 并将这些下标处的值减 1。

数组将变为 [4, 2, 1, 0]。

对于 i = 1：

选择下标子集 [0, 1, 2] 并将这些下标处的值减 1。

数组将变为 [3, 1, 0, 0]，这不是一个零数组。

提示：

1 <= nums.length <= 105

0 <= nums[i] <= 105

1 <= queries.length <= 105

queries[i].length == 2

0 <= li <= ri < nums.length

# 分析

要解决“处理所有查询后能否将数组转换为零数组”的问题，核心思路是计算每个元素被查询覆盖的次数，并验证该次数是否至少等于元素的初始值。通过高效统计每个下标被查询区间覆盖的次数，可判断是否满足条件。

解题思路

1、核心观察：

每个查询[l, r]可对区间`[l, r]`内的元素各“分配一次减1机会”。处理所有查询后，若每个元素被覆盖的次数≥其初始值，则数组可变为零数组（每次查询选择需要减1的元素即可）。

2、高效统计覆盖次数：

直接遍历每个查询并更新区间内所有元素的覆盖次数（O(n×q)）会超时（n和q均可达1e5）。需使用差分数组优化区间更新：

- 差分数组diff：diff[l] += 1表示区间[l, ...]增加1次覆盖，diff[r+1] -= 1表示区间[r+1, ...]减少1次覆盖。

- 通过差分数组的前缀和，可在O(n + q)时间内计算每个下标被覆盖的总次数。

3、验证条件：

对每个下标i，若覆盖次数cnt[i] < nums[i]，则无法将该元素减为0，返回false；否则返回true。

代码：

class Solution {

public:

bool isZeroArray(vector<int>& nums, vector<vector<int>>& queries) {

int n = nums.size();

// 差分数组：用于高效统计每个下标被查询覆盖的次数

vector<int> diff(n + 1, 0); // 大小n+1避免r+1越界

// 处理所有查询，更新差分数组

for (auto& q : queries) {

int l = q[0];

int r = q[1];

diff[l]++; // 区间[l, ...]覆盖次数+1

if (r + 1 < n) {

diff[r + 1]--; // 区间[r+1, ...]覆盖次数-1（若r+1在数组内）

}

}

// 计算前缀和，得到每个下标被覆盖的总次数cnt

int cnt = 0;

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

cnt += diff[i]; // 前缀和即当前下标i的覆盖次数

// 若覆盖次数小于初始值，无法减为0

if (cnt < nums[i]) {

return false;

}

}

return true;

}

};

代码解释：

1、差分数组初始化：

diff数组大小为n+1，用于处理r = n-1时r+1 = n的边界情况（避免越界）。

2、处理查询更新差分数组：

对每个查询[l, r]：

- diff[l]++：标记从l开始的区间覆盖次数增加1。

- diff[r+1]--：标记从r+1开始的区间覆盖次数减少1（若r+1 < n）。

3、计算覆盖次数并验证：

通过前缀和计算cnt（当前下标i的总覆盖次数）：

- 若cnt < nums[i]：该元素无法被减为0，返回false。

- 遍历结束后所有元素均满足cnt ≥ nums[i]，返回true。

复杂度分析

- 时间复杂度：O(n + q)，其中n是数组长度，q是查询数量。

处理所有查询更新差分数组：O(q)。

计算前缀和并验证每个元素：O(n)。

- 空间复杂度：O(n)，差分数组diff的大小为n+1。