# 题目

给你一个整数数组 arr 。请你返回和为 奇数 的子数组数目。

由于答案可能会很大，请你将结果对 10^9 + 7 取余后返回。

示例 1：

输入：arr = [1,3,5]

输出：4

解释：所有的子数组为 [[1],[1,3],[1,3,5],[3],[3,5],[5]] 。

所有子数组的和为 [1,4,9,3,8,5].

奇数和包括 [1,9,3,5] ，所以答案为 4 。

示例 2 ：

输入：arr = [2,4,6]

输出：0

解释：所有子数组为 [[2],[2,4],[2,4,6],[4],[4,6],[6]] 。

所有子数组和为 [2,6,12,4,10,6] 。

所有子数组和都是偶数，所以答案为 0 。

示例 3：

输入：arr = [1,2,3,4,5,6,7]

输出：16

示例 4：

输入：arr = [100,100,99,99]

输出：4

示例 5：

输入：arr = [7]

输出：1

提示：

1 <= arr.length <= 10^5

1 <= arr[i] <= 100

# 分析

## 方法一：前缀和

思路：

可以使用前缀和的思想来解决这个问题。首先计算出数组 arr 的前缀和数组 prefixSum，然后遍历 prefixSum，对于每个前缀和，判断其奇偶性，统计奇数前缀和出现的次数和偶数前缀和出现的次数。最后计算奇数前缀和出现次数乘以偶数前缀和出现次数，即为所求结果。由于结果可能很大，需要对结果取模1e9+7。

代码：

class Solution {

public:

int numOfSubarrays(vector<int>& arr) {

const int MOD = 1e9 + 7;

int oddSumCount = 0, evenSumCount = 1, prefixSum = 0, result = 0;

for (int num : arr) {

// 计算前缀和

prefixSum += num;

if (prefixSum % 2 == 0) {

// 如果前缀和为偶数，则当前前缀和可以由前一个奇数前缀和加上当前数字得到

// 因此将奇数前缀和的数量加到结果中，并增加偶数前缀和的数量

result = (result + oddSumCount) % MOD;

evenSumCount++; // 偶数前缀和

} else {

// 如果前缀和为奇数，则当前前缀和可以由前一个偶数前缀和加上当前数字得到

// 因此将偶数前缀和的数量加到结果中，并增加奇数前缀和的数量

result = (result + evenSumCount) % MOD;

oddSumCount++;

}

}

return result;

}

};