# 题目

给你一个正整数数组arr，请你计算所有可能的奇数长度子数组的和。

子数组定义为原数组中的一个连续子序列。

请你返回arr中所有奇数长度子数组的和 。

示例 1：

输入：arr = [1,4,2,5,3]

输出：58

解释：所有奇数长度子数组和它们的和为：

[1] = 1

[4] = 4

[2] = 2

[5] = 5

[3] = 3

[1,4,2] = 7

[4,2,5] = 11

[2,5,3] = 10

[1,4,2,5,3] = 15

我们将所有值求和得到 1 + 4 + 2 + 5 + 3 + 7 + 11 + 10 + 15 = 58

示例 2：

输入：arr = [1,2]

输出：3

解释：总共只有 2 个长度为奇数的子数组，[1] 和 [2]。它们的和为 3 。

示例 3：

输入：arr = [10,11,12]

输出：66

提示：

1 <= arr.length <= 100

1 <= arr[i] <= 1000

进阶：

你可以设计一个 O(n) 时间复杂度的算法解决此问题吗？

# 分析

## 方法一：暴力法

思路：

代码：

class Solution {

public:

int sumOddLengthSubarrays(vector<int>& arr) {

int sum = 0;

int n = arr.size();

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

for (int length = 1; start + length <= n; length += 2) {

int end = start + length - 1;

for (int i = start; i <= end; i++) {

sum += arr[i];

}

}

}

return sum;

}

};

## 方法二：模拟

思路：

通过计算每个元素在奇数长度子数组中出现的次数来解决这个问题。考虑一个元素 arr[i]，它在奇数长度子数组中出现的次数取决于它左边和右边的元素个数。假设 arr[i] 的下标为 i，数组长度为 n，则 arr[i] 左边有 i 个元素，右边有 n-i-1 个元素。所以 arr[i] 在奇数长度子数组中出现的次数为 (i+1)\*(n-i)。

因此，你可以遍历数组 arr，计算每个元素在奇数长度子数组中出现的次数，并累加得到结果。

代码：

class Solution {

public:

int sumOddLengthSubarrays(vector<int>& arr) {

int n = arr.size();

int sum = 0;

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

int left\_count = i + 1;

int right\_count = n - i;

int total\_count = left\_count \* right\_count;

int odd\_count = total\_count / 2 + total\_count % 2;

sum += odd\_count \* arr[i];

}

return sum;

}

};

## 方法三：前缀和