# 题目

给你一个整数数组 arr，只有可以将其划分为三个和相等的 非空 部分时才返回 true，否则返回 false。

形式上，如果可以找出索引 i + 1 < j 且满足 (arr[0] + arr[1] + ... + arr[i] == arr[i + 1] + arr[i + 2] + ... + arr[j - 1] == arr[j] + arr[j + 1] + ... + arr[arr.length - 1]) 就可以将数组三等分。

示例 1：

输入：arr = [0,2,1,-6,6,-7,9,1,2,0,1]

输出：true

解释：0 + 2 + 1 = -6 + 6 - 7 + 9 + 1 = 2 + 0 + 1

示例 2：

输入：arr = [0,2,1,-6,6,7,9,-1,2,0,1]

输出：false

示例 3：

输入：arr = [3,3,6,5,-2,2,5,1,-9,4]

输出：true

解释：3 + 3 = 6 = 5 - 2 + 2 + 5 + 1 - 9 + 4

提示：

3 <= arr.length <= 5 \* 10^4

-10^4 <= arr[i] <= 10^4

# 分析

## 方法一：一次遍历

思路：

要判断一个数组能否被划分为三个和相等的部分，可以按照以下步骤进行：

1、计算数组的总和 sum，如果 sum 不能被 3 整除，直接返回 false。

2、初始化两个指针 left 和 right 分别指向数组的两端，初始化两个变量 partSum 和 count，分别表示当前部分的和和已经满足条件的部分数。

3、从左到右遍历数组，累加当前元素到 partSum 中，并判断是否等于 sum / 3，如果等于，则将 count 加 1，同时重置 partSum 为 0。

4、如果 count 达到 2，说明已经找到了两个满足条件的部分，那么剩下的部分必然也满足条件，返回 true；否则，继续遍历直到结束，最终返回 false。

代码：

class Solution {

public:

bool canThreePartsEqualSum(vector<int>& arr) {

int sum = accumulate(arr.begin(), arr.end(), 0);

if (sum % 3 != 0) {

return false;

}

int partSum = 0;

int count = 0;

for (int i = 0; i < arr.size(); ++i) {

partSum += arr[i];

if (partSum == sum / 3) {

count++;

partSum = 0;

}

if (count == 2) {

return true;

}

}

return false;

}

};