1062 Partition Array Into Three Parts With Equal Sum (link)

Description

Given an array of integers `arr`, return `true` if we can partition the array into three **non-empty** parts with equal sums.

Formally, we can partition the array if we can find indexes `i + 1 < j` with `(arr[0] + arr[1] + ... + arr[i] == arr[i + 1] + arr[i + 2] + ... + arr[j - 1] == arr[j] + arr[j + 1] + ... + arr[arr.length - 1])`

Example 1:

```
Input: arr = [0,2,1,-6,6,-7,9,1,2,0,1]
Output: true
Explanation: 0 + 2 + 1 = -6 + 6 - 7 + 9 + 1 = 2 + 0 + 1
```

Example 2:

```
Input: arr = [0,2,1,-6,6,7,9,-1,2,0,1]
Output: false
```

Example 3:

```
Input: arr = [3,3,6,5,-2,2,5,1,-9,4]
Output: true
Explanation: 3 + 3 = 6 = 5 - 2 + 2 + 5 + 1 - 9 + 4
```

Constraints:

- `3 <= arr.length <= 5 * 10^4`
- `-10^4 <= arr[i] <= 10^4`

(scroll down for solution)

Solution

Language: cpp

Status: Accepted

```cpp
#include <vector>
using namespace std;

class Solution {
public:
    bool canThreePartsEqualSum(vector<int>& arr) {
        int totalSum = 0;
        for (int num : arr) {
            totalSum += num;
        }

        if (totalSum % 3 != 0) {
            return false;
        }

        int target = totalSum / 3;
        int currentSum = 0, partsFound = 0;

        for (int i = 0; i < arr.size(); ++i) {
            currentSum += arr[i];
            if (currentSum == target) {
                partsFound++;
                currentSum = 0;
            }
        }


        return partsFound >= 3;
    }
};
```