Question 3: Finding Equilibrium Index in an Array

Problem

Find the equilibrium index of an array. An equilibrium index is an index such that the sum of elements at lower indices equals the sum of elements at higher indices.

Algorithm

1. Calculate the total sum of all elements in the array.
2. Initialize leftSum = 0.
3. Iterate through the array: a. Calculate rightSum = totalSum - leftSum - current element. b. If leftSum equals rightSum, return the current index. c. Update leftSum by adding the current element to it.
4. If no equilibrium index is found, return -1.

Program Implementation

cpp

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

int findEquilibriumIndex(const vector<int>& arr) {
    int totalSum = 0;
    for (int num : arr) {
        totalSum += num;
    }
    
    int leftSum = 0;
    for (int i = 0; i < arr.size(); i++) {
        int rightSum = totalSum - leftSum - arr[i];
        
        if (leftSum == rightSum) {
            return i;
        }
        
        leftSum += arr[i];
    }
    
    return -1;
}

int main() {
    vector<int> arr1 = {1, 7, 3, 6, 5, 6};
    cout << "Equilibrium index for array 1: " << findEquilibriumIndex(arr1) << endl;
    
    vector<int> arr2 = {1, 2, 3, 4, 5};
    cout << "Equilibrium index for array 2: " << findEquilibriumIndex(arr2) << endl;
    
    vector<int> arr3 = {-7, 1, 5, 2, -4, 3, 0};
    cout << "Equilibrium index for array 3: " << findEquilibriumIndex(arr3) << endl;
    
    return 0;
}
```

Time Complexity

- O(n) where n is the size of the array
- We make two passes through the array: one to calculate the total sum and another to find the equilibrium index

Space Complexity

- O(1) as we only use a constant amount of extra space regardless of input size

Example Explanation

Let's trace through the example using the array [1, 7, 3, 6, 5, 6]:

1. Total sum = 1 + 7 + 3 + 6 + 5 + 6 = 28
2. Iteration through the array:
    - At index 0:
        - leftSum = 0
        - rightSum = 28 - 0 - 1 = 27
        - Not an equilibrium index
        - Update leftSum = 0 + 1 = 1
    - At index 1:
        - leftSum = 1
        - rightSum = 28 - 1 - 7 = 20
        - Not an equilibrium index
        - Update leftSum = 1 + 7 = 8
    - At index 2:
        - leftSum = 8
        - rightSum = 28 - 8 - 3 = 17
        - Not an equilibrium index
        - Update leftSum = 8 + 3 = 11
    - At index 3:
        - leftSum = 11
        - rightSum = 28 - 11 - 6 = 11
        - leftSum equals rightSum, so index 3 is an equilibrium index
        - Return 3

The function returns 3, which is indeed the equilibrium index of the array [1, 7, 3, 6, 5, 6] because the sum of elements before index 3 (1+7+3=11) equals the sum of elements after index 3 (5+6=11).