LeetCode 724 — Find Pivot Index

■ Question Summary
You're given an integer array nums. You must find the pivot index, i.e., the index i such that:
sum(nums[0 ... i-1]) == sum(nums[i+1 ... n-1])
If no such index exists → return -1.
If multiple pivots exist → return the leftmost one.

■ Example
Input: nums = [1,7,3,6,5,6]
Output: 3
Left sum = 1 + 7 + 3 = 11, Right sum = 5 + 6 = 11

■ Brute Force Approach — O(n²)
For each index i:
- Calculate left sum (before i)
- Calculate right sum (after i)
- Compare them

Dry Run
nums = [1,7,3,6,5,6]
For i=3: Left=11, Right=11 ■

C++ Code (Brute Force)
```
#include <bits/stdc++.h>
using namespace std;
int pivotIndex(vector<int>& nums) {
    int n = nums.size();
    for(int i=0; i<n; i++) {
        int left = accumulate(nums.begin(), nums.begin()+i, 0);
        int right = accumulate(nums.begin()+i+1, nums.end(), 0);
        if(left == right) return i;
    }
    return -1;
}
int main(){
    vector<int> nums = {1,7,3,6,5,6};
    cout << pivotIndex(nums);
}
```

```
class Main {
    public static int pivotIndex(int[] nums) {
        for(int i=0; i<nums.length; i++){
            int left=0, right=0;
            for(int j=0;j<i;j++) left += nums[j];
            for(int j=i+1;j<nums.length;j++) right += nums[j];
            if(left==right) return i;
        }
        return -1;
    }
    public static void main(String[] args){
        int[] nums = {1,7,3,6,5,6};
        System.out.println(pivotIndex(nums));
    }
}
def pivotIndex(nums):
```

```
    for i in range(len(nums)):
        left = sum(nums[:i])
        right = sum(nums[i+1:])
        if left == right:
            return i
    return -1

print(pivotIndex([1,7,3,6,5,6]))
```

■ **Optimized Approach — Using Prefix Sum (O(n))**
Idea:
1. Compute total sum
2. Keep a running left sum
3. For each i: right_sum = total - left_sum - nums[i]
If left_sum == right_sum → return i
else left_sum += nums[i]

Dry Run
nums = [1,7,3,6,5,6], total=28
i=3 → left=11, right=11 ■ Pivot=3

```
#include <bits/stdc++.h>
using namespace std;
int pivotIndex(vector<int>& nums) {
    int total = accumulate(nums.begin(), nums.end(), 0);
    int left = 0;
    for(int i=0; i<nums.size(); i++) {
        int right = total - left - nums[i];
        if(left == right) return i;
        left += nums[i];
    }
    return -1;
}
int main(){
    vector<int> nums = {1,7,3,6,5,6};
    cout << pivotIndex(nums);
}
class Main {
    public static int pivotIndex(int[] nums) {
        int total = 0;
        for(int num : nums) total += num;
        int left = 0;
        for(int i=0;i<nums.length;i++){
            int right = total - left - nums[i];
            if(left == right) return i;
            left += nums[i];
        }
        return -1;
    }
    public static void main(String[] args){
        int[] nums = {1,7,3,6,5,6};
        System.out.println(pivotIndex(nums));
    }
}
def pivotIndex(nums):
    total = sum(nums)
    left = 0
```

```
    for i, num in enumerate(nums):
        if left == total - left - num:
            return i
        left += num
    return -1

print(pivotIndex([1,7,3,6,5,6]))
```

■ Time & Space Complexity
Brute Force: O(n²) time, O(1) space
Prefix Sum: O(n) time, O(1) space

■ Example 2
Input: [1,2,3]
Output: -1 (no pivot index)