HOMEWORK DAY: 11

14 September 2023 08:19

HOMEWORK OF CLASS: 01 DATE:13/09/2023
https://www.linkedin.com/in/manojofficialmj

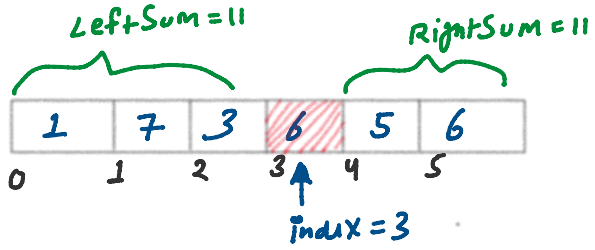
1. Find pivot element (Leetcode-724)

Example 1:
Input: nums = [1,7,3,6,5,6]
Output: 3
Explanation:
The pivot index is 3.
Left sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11
Right sum = nums[4] + nums[5] = 5 + 6 = 11

Example 2:
Input: nums = [1,2,3]
Output: -1
Explanation:
There is no index that satisfies the conditions in the problem statement.

DRY RUN

Example 1:



APPROACH PSEUDO CODE:
Step 01: find total sum of array as right sum
Step 02: subtract element one by one from right sum until left sum is equal to right sum
Step 03: return the index which terminate the loop

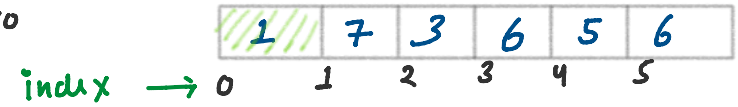
(1)

TotalSum = 1+7+3+6+5+6

RightSum = 28

(2)

iteration:0



RightSum = 28
LeftSum = 0
index = 0

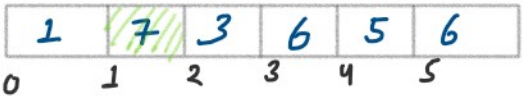
↳ RightSum = RightSum − nums[index]
= 28 − 1
= 27

↳ (RightSum == LeftSum)
27 == 0 ✗ FAIL

↳ LeftSum = LeftSum + nums[index]
= 0 + 1
= 1

iteration:1

Situation: 1



RightSum = 27
LeftSum = 1
index = 1

↳ RightSum = RightSum − nums[index]
= 27 − 7
= 20

↳ (RightSum == LeftSum)
20 == 1 ✗ FAIL

↳ LeftSum = LeftSum + nums[index]
= 1 + 7
= 8

Situation: 2

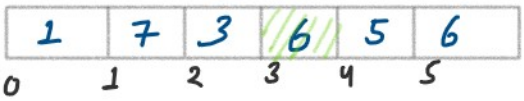
RightSum = 20
LeftSum = 8
index = 2

↳ RightSum = RightSum − nums[index]
= 20 − 3
= 17

↳ (RightSum == LeftSum)
17 == 8 ✗ FAIL

↳ LeftSum = LeftSum + nums[index]
= 8 + 3
= 11

Situation: 3



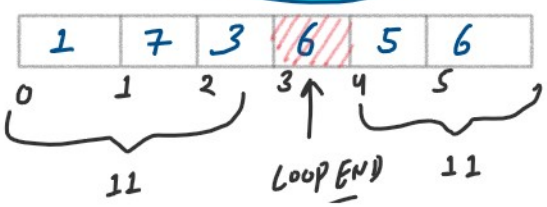
RightSum = 17
LeftSum = 11
index = 3

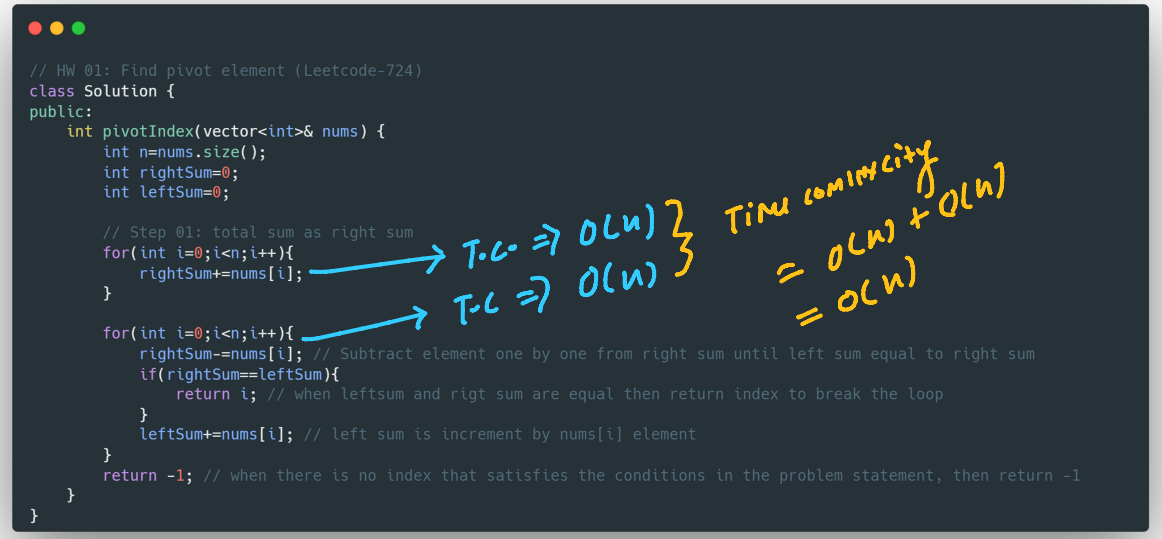
↳ RightSum = RightSum − nums[index]
= 17 − 6
= 11

↳ (RightSum == LeftSum)
11 == 11 ✓ TRUE

↳ return index

Output : 3




// HW 01: Find pivot element (Leetcode-724)
class Solution {
public:
int pivotIndex(vector<int>& nums) {
int n=nums.size();
int rightSum=0;
int leftSum=0;
// Step 01: total sum as right sum
for(int i=0;i<n;i++){
rightSum+=nums[i];
}
for(int i=0;i<n;i++){
rightSum-=nums[i]; // Subtract element one by one from right sum until left sum equal to right sum
if(rightSum==leftSum){
return i; // when leftsum and rigt sum are equal then return index to break the loop
}
leftSum+=nums[i]; // left sum is increment by nums[i] element
}
return -1; // when there is no index that satisfies the conditions in the problem statement, then return -1
}
}
T.C => O(n)
T.C => O(n)
Time complexity
= O(n) + O(n)
= O(n)