**Subarray with given sum :-**

Easy Accuracy: 16.5% Submissions: 1.3M Points: 2

Given an unsorted array **A**of size **N** that contains only positive integers, find a continuous sub-array that adds to a given number **S**and return the left and right index(**1-based indexing**) of that subarray.

In case of multiple subarrays, return the subarray indexes which come first on moving from left to right.

**Note**:- You have to return an ArrayList consisting of two elements left and right. In case no such subarray exists return an array consisting of element **-1**.

**Example 1:**

**Input:**

N = 5, S = 12

A[] = {1,2,3,7,5}

**Output:** 2 4

**Explanation:** The sum of elements

from 2nd position to 4th position

is 12.

**Example 2:**

**Input:**

N = 10, S = 15

A[] = {1,2,3,4,5,6,7,8,9,10}

**Output:** 1 5

**Explanation:** The sum of elements

from 1st position to 5th position

is 15.

**Your Task:**  
You don't need to read input or print anything. The task is to complete the function **subarraySum**() which takes arr, N, and S as input parameters and returns an **ArrayList**containing the **starting**and **ending**positions of the first such occurring subarray from the left where sum equals to S. The **two indexes in the array** should be according to **1-based indexing**. If no such subarray is found, return an array consisting of only one element that is -1.

**Expected Time Complexity:**O(N)  
**Expected Auxiliary Space:**O(1)

**Constraints:**  
1 <= N <= 105  
1 <= Ai <= 109  
0<= S <= 109

**Code :-**

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution

{

public:

//Function to find a continuous sub-array which adds up to a given number.

vector<int> subarraySum(vector<int>arr, int n, long long s){

//cout<<"["<<arr[0]<<","<<arr[1]<<"]";

int startans=-1, endans=-1;

long long sum=arr[0];

int start=0, end=0;

while(end < n){

// target is found

if(sum == s){

startans = start;

endans = end;

break;

}

// current sum is greater than target

else if(sum > s){

sum -= arr[start];

++start;

}

// current sum is lesser than target

else{

++end;

sum += arr[end];

}

}

if(end != n and end>=start){

return {startans+1, endans+1};

}

return {-1};

}

};

//{ Driver Code Starts.

int main()

{

int t;

cin>>t;

while(t--)

{

int n;

long long s;

cin>>n>>s;

vector<int>arr(n);

// int arr[n];

const int mx = 1e9;

for(int i=0;i<n;i++)

{

cin>>arr[i];

}

Solution ob;

vector<int>res;

res = ob.subarraySum(arr, n, s);

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

cout<<res[i]<<" ";

cout<<endl;

}

return 0;

}

// } Driver Code Ends

**T.C :- O(2\*N)**

**S.C :- O(N)**