6th JAN 2023

FIRST :-

Convert array into Zig-Zag fashion :: Basic

Given an array **arr** of distinct elements of size **N**, the task is to rearrange the elements of the array in a zig-zag fashion so that the converted array should be in the below form:

arr[0] < arr[1] > arr[2] < arr[3] > arr[4] < . . . . arr[n-2] < arr[n-1] > arr[n].

NOTE: If your transformation is correct, the output will be 1 else the output will be 0.

Example 1:

```
Input:
N = 7
Arr[] = {4, 3, 7, 8, 6, 2, 1}
Output: 3 7 4 8 2 6 1
Explanation: 3 < 7 > 4 < 8 > 2 < 6 > 1
```

Example 2:

```
Input:
N = 4
Arr[] = {1, 4, 3, 2}
Output: 1 4 2 3
Explanation: 1 < 4 > 2 < 3
```

Your Task:

You don't need to read input or print anything. Your task is to complete the function **zigZag()** which takes the array of integers **arr** and **n** as parameters and returns void. You need to modify the array itself.

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

Constraints:
$1 <= N <= 10^6$
$0 <= Arr_i <= 10^9$

CODE SECTION:-

```
class Solution {
  public:
    // Program for zig-zag conversion of array
    void zigZag(int arr[], int n) {
        // code here
        int i,j,k;

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

            if(i%2==0){

                if(arr[i]>arr[i+1]){
                    swap(arr[i],arr[i+1]);
                }
            }
            else{
                if(arr[i]<arr[i+1]){
                    swap(arr[i],arr[i+1]);
                }
            }
        }



    }
};
```

SECOND:-

Subarray with given sum : : Easy

Given an unsorted array **A** of size **N** that contains only non-negative integers, find a continuous sub-array which 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 comes first on moving from left to right.

Note:- Both the indexes in the array should be according to **1-based indexing.** 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)

CODE SECTION:-

```
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)
    {
        // Your code here
        if(s==0){
            return {-1};
        }

      int i=0;
      int j=0;
      long long currsum=0;
      vector<int> v;
      while(i<n && j<=n){

          if(currsum<s){
              currsum+=arr[j];
               j++;
          }
          else if(currsum>s){
              currsum-=arr[i];
              i++;
          }
          else {
              v.push_back(i+1);
              v.push_back(j);
              return v;
          }
      }
      v.push_back(-1);
      return v;

    }

};
```

-: Done for the today :-