Given an array of size “N” , you need to find the shortest subarray such that the sum of first and last element is K

Sample Testcase :

[4 , 6 , 7 ,8 , 1 , 7 , 8 , 9 , 10] k = 8

Output : 2 [1 , 7]

Brute Force :

class Solution {

    public int Ksum(int [] nums , int k) {

        int n =  nums.length;

        int min = n + 1;

        for(int i = 0 ; i < n ; i++){

            for(int j = i + 1 ; j < n ; j++){

                if(nums[i] + nums[j] == k){

                    min = Math.min(min , j - i);

                }

            }

        }

        return min;

    }

}

Time Complexcity : O (N \* N)

Space Complexcity : O (1)

Optimized Approach :

Step 1 : Store all the elements in the hasMap along with its index . 🡪 O (N)

Step 2 : Run a for loop from 0 to n and for each iteration check whether an element k – nums[i] exists in the array.

Step 3 : If so , calculate the length by , | map.get(elment) – I | and get the minimum of it.

Step 4 : Return the minimum length.

class Solution {

    public int Ksum(int [] nums , int k) {

        int n =  nums.length;

        int min = n + 1;

        Map<Integer , Integer> map = new HashMap<>();

        for(int i = 0 ; i < n ; i++){

            int req = k - nums[i];

            if(map.contains(req)){

                min = Math.min(min , i - map.get(req));

            }

            map.put(nums[i] , i);

        }

        return min + 1;

    }

}

Time Complexcity : O (N)

Space Complexcity : O (N)

Follow Up :

Find the longest subarray whose first and last index sunm is K

Soln : Instaed of updating the map each time , update the map only if it is not found.