Assessment -10 [Power Seeds]

By- Joydeep Biswas

Difficulty:- Easy

Prerequisites:- Dual Pointer technique [or Hash Map].

Problem Understanding:- Simply the problem says find out the pair of elements which has same sum as **K** [you are given a array of **N** elements]. Then just print the count of the valid pairs. One main thing is we can't reuse one element to form a pair.

Example:- suppose we have **1 1 1 1** [these four element] and K is given 2. So we need to find the pair of element whose sum is equal to 2. Now you can form only two valid pairs whose sum is 2 because you can't reuse any element. So the valid pairs are **[1,1] & [1,1]** you can see we are not reuse any element.

N.B:- if you reuse the elements then there will be more than 2 pairs because each 1 can form pair with any another 1.

Solution Approach:- After understand the problem statement the very first idea which hits out mind is that using brute force. Simply check all the elements of the array using two for loop and maintain a visited array so that no element comes twice. But this approach will not work because the constraints is not satisfying **$O(n^2)$** solution. Where **N is 10^5**.

Approach for 100 point:- So, by looking at the time constraints we have to solve this problem at maximum of O(n log n) time. For that we need to follow a very important approach called " **Dual Pointer Technique** ". Its not so hard as its sounds. Basically here what we do, we maintain two pointer and check until it overlaps [or depends on the condition].

Explanation:-

To solve this problem

1. Sort the array
2. Initialize two variable **L** and **R**. where **L=0** and **R=n-1**
3. Traverse the array until it overlaps or until **(L < R)**;
4. **While(L<R)** just count the number of pair having the sum equal to **K.**
5. Also check each time that if the sum of the pair is greater than **K** then decrement the **R** by 1.
6. And if the sum of the pair is less than **K** increment the **L** by 1.
7. And If the sum is equal to **K** then increment the **L** by 1 and decrement the **R** by 1.

Now simply print the count.

If you still have some doubts then you can read this part.Now you can see that, when you sort the array all the elements are in proper order. So when we check for equal sum to K and it fails means that those two pointer L & R are not in correct position, simply move those pointer. But the question is how? To understand this think about a number less than the desire number in a sorted array. Defiantly you will find it on left side of the desire number in the array, because it is smaller than your desire number and the array is sorted in accessing order. Same for the bigger number than you desire number should be in right side of your desire number in the array. With this simple concept we move the pointers and set them on a desire position. If you still confuse then see this example.

Example:-

4 6

1 5 7 1

In this array 1 5 7 1 we have to find the number of pair whose sum is equal to 6

1. After sorting the array will be 1 1 5 7
2. Initialize two pointer L = 0 and R = n-1 = 3;
3. Running loop while(L<R)

```
1 1 5 7
^     ^
L     R            L<R satisfies
```

Checking if the pair arr[L] + arr[R] = 6 or not .

If not then weather arr[L]+arr[R] > 6 or < 6

If > 6 then decrement the R by 1

```
1 1 5 7
^   ^
L   R              L<R satisfies
```

Now it satisfies so count=1 and increment L by 1 and decrement R by 1

```
1 1 5 7
  ^
  LR               L<R not satisfies loop break
```

So out answer is 1.

Complexity:- total complexity is O(n log n)

N.B:- You can solve this problem by HashMap using O(n) time. I"ll discuss this later. Today's goal is to understand dual pointer concept using a simple problem.

Assessment -10 [Power Seeds]

Code:-

```
//// By- Joydeep Biswas [27/04/2020]

//
//solution for 100 points
//
//O(nlogn) complexity
//
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.*;
class TestClass {
    public static void main(String args[] ) throws Exception {
       BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
        int t = Integer.parseInt(reader.readLine());
        while(t-->0) {
            String[] s = reader.readLine().split(" ");
            int n = Integer.parseInt(s[0]);
            int k = Integer.parseInt(s[1]);
            int[] arr = new int[n];
            String[] input = reader.readLine().split(" ");
            for(int i=0;i<n;i++) {
                arr[i] = Integer.parseInt(input[i]);
            }
            Arrays.sort(arr);
            int l=0;
            int r=n-1;
            int count=0;
            while(l<r) {
                if(Math.abs(arr[l]+arr[r])==k) {
                    count++;
                    l++;
                    r--;
                }
                else if(Math.abs(arr[l]+arr[r])>k) {
                    r--;
                }
                else if(Math.abs(arr[l]+arr[r])<k) {
                    l++;
                }
            }
            System.out.println(count);
        }

    }
}
```