INTERNSHALA TRAININGS

Career Ki Guarantee

DATA STRUCTURE AND ALGORITHMS
ASSIGNMENT – I [ARRAYS]

Problem 1:

Kids with the greatest number of candies

There are n kids with candies. You are given an integer array candies, where each candies[i] represents the number of candies the ith kid has, and an integer extraCandies, denoting the number of extra candies that you have. Return a boolean array result of length n, where result[i] is true if, after giving the ith kid all the extraCandies, they will have the greatest number of candies among all the kids, or false otherwise.

Example 1:

Input: candies = [2,3,5,1,3], extraCandies = 3

Output: [true,true,true,false,true]

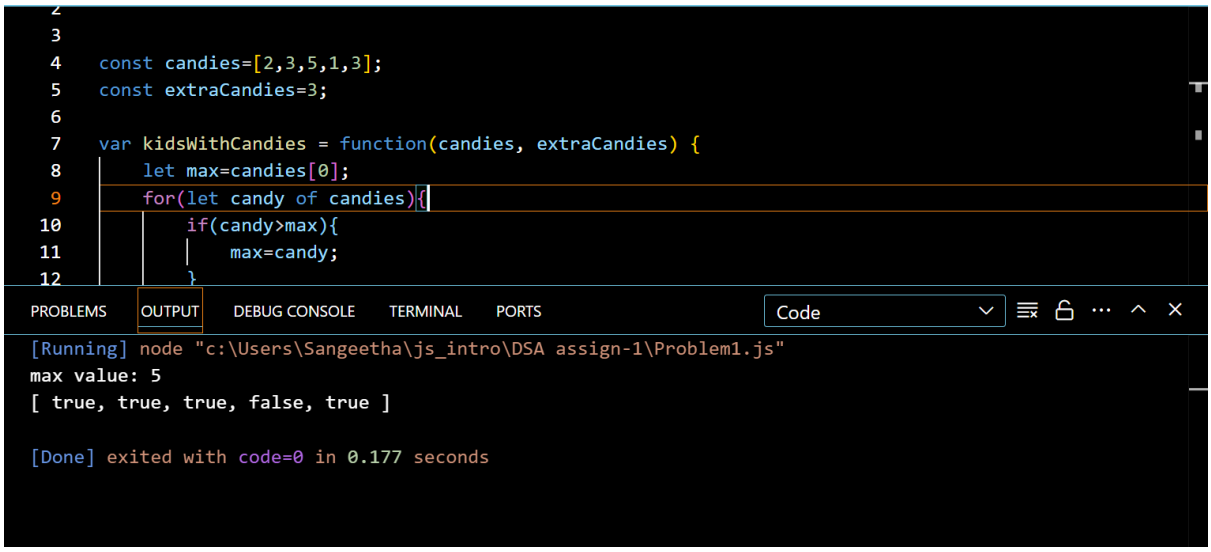
Question link:

https://leetcode.com/problems/kids-with-the-greatest-number-of-candies/description/

Code:

```
var kidsWithCandies = function(candies, extraCandies) {
    let max=candies[0];
    for(let candy of candies){
        if(candy>max){
            max=candy;
        }
    }
    console.log("Max value:",max);
    return candies.map(candy=>candy+extraCandies>=max);


};
```

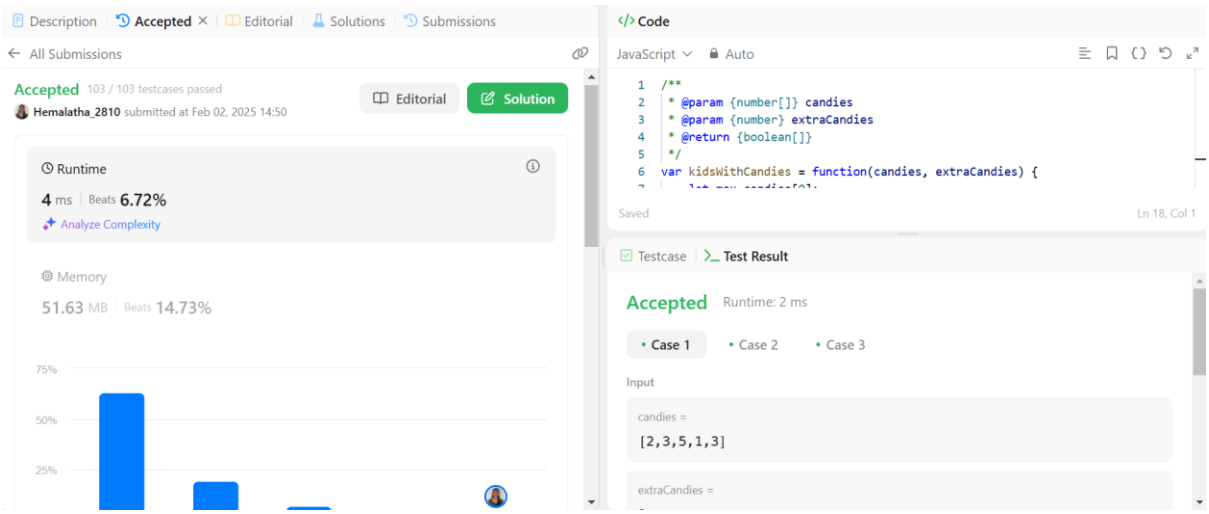
Output:



Leet-code submission link:

https://leetcode.com/problems/kids-with-the-greatest-number-of-candies/submissions/1528477730/

Screenshot:



Conclusion:

Time complexity: O(n)

- For loop iterates the candies to find max element in the array→ O(n)
- Map function also iterates over the candies array →O(n)
- Finally,

$$O(n) + O(n) = O(n)$$

Space complexity: O(n)

- The map function requires n space.

Problem-2:

Count number of pairs with absolute difference K

Given an integer array nums and an integer k, return the number of pairs (i, j) where i < j such that |nums[i] - nums[j]| == k. The value of |x| is defined as:

x if x >= 0.

-x if x < 0.

Example-1:

Input: nums = [1,2,2,1], k = 1

Output: 4

Question link:

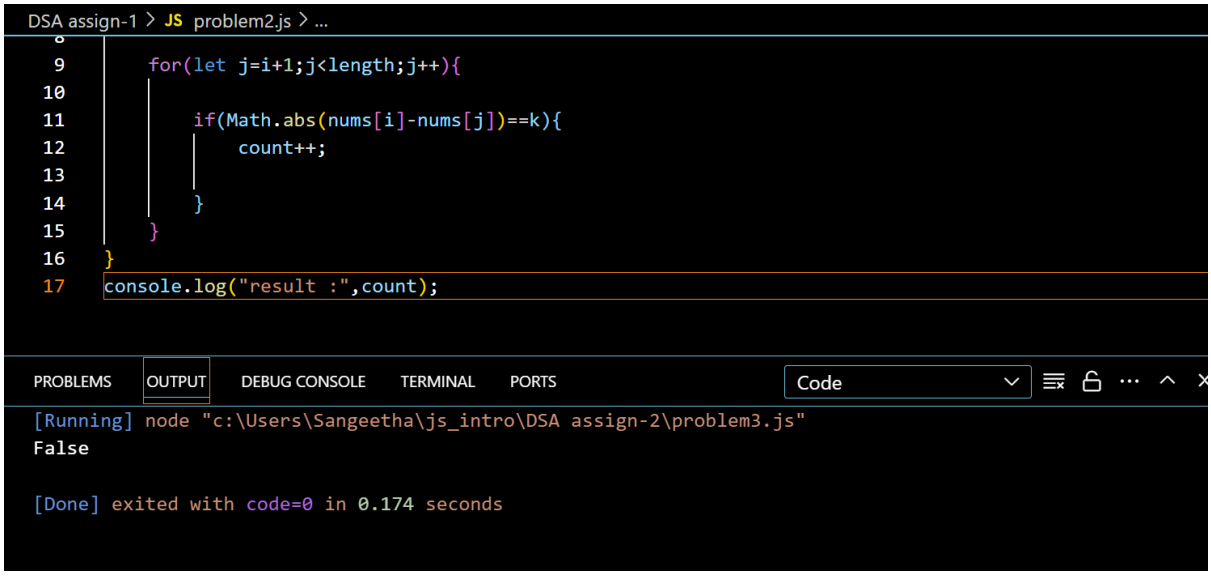
https://leetcode.com/problems/count-number-of-pairs-with-absolute-difference-k/description/

Code:

```
const nums = [1,2,2,1];

const k=1;

let length=nums.length;

let count=0;

for(let i=0;i<length;i++){
```

```
    for(let j=i+1;j<length;j++){
        if(Math.abs(nums[i]-nums[j])==k){
            count++;


        }
    }
}
console.log("result :",count);
```

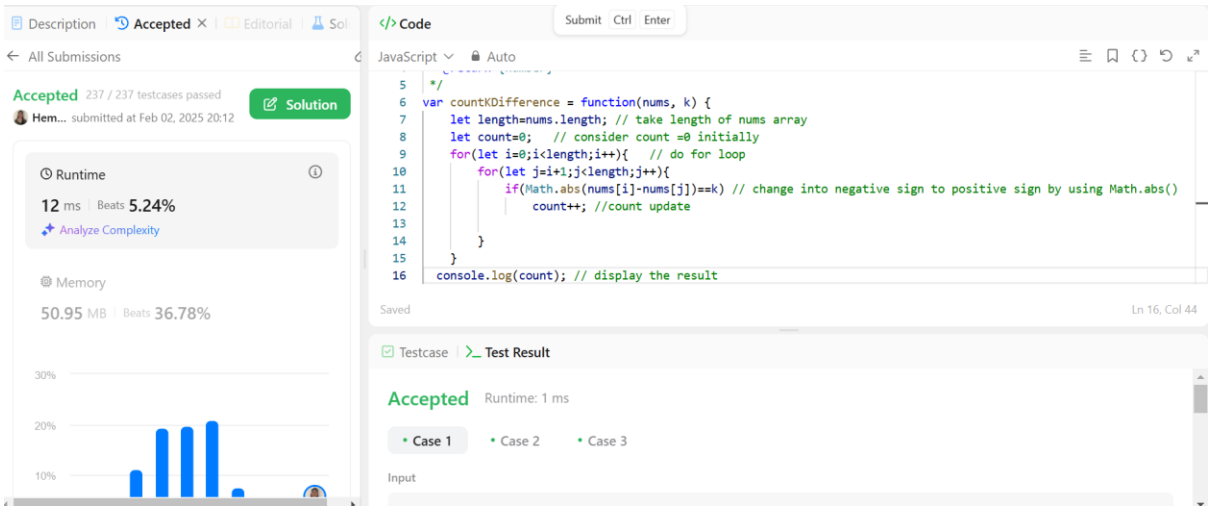
Output:



Leet-code submission link:

https://leetcode.com/problems/count-number-of-pairs-with-absolute-difference-k/submissions/1528734140/

Screenshot:



Conclusion:

Time complexity: $O(n^2)$

- Nested loops takes $O(n^2)$ time complexity. Math.abs() takes O(1)time complexity.

Space complexity: O(1)

- Variables takes only O(1) space complexity. The nums array did not take extra space.

Problem 3:

Find common elements between two arrays

You are given two integer arrays nums1 and nums2 of sizes n and m, respectively. Calculate the following values:

Answer1 : the number of indices i such that nums1[i] exists in nums2.

Answer2 : the number of indices i such that nums2[i] exists in nums1.

Return [answer1,answer2].

Example 1:

Input: nums1 = [2,3,2], nums2 = [1,2]

Output: [2,1]

Question link:

https://leetcode.com/problems/find-common-elements-between-two-arrays/description/

Code:

```
var findIntersectionValues = function(nums1, nums2) {
    let count1=0;
    let count2=0;
    let newNum1=[...new Set(nums1)];
```

```
    let newNum2=[...new Set(nums2)];
    for(let i=0;i<nums1.length;i++){
        for(let j=0;j<newNum2.length;j++){
            if(nums1[i]==newNum2[j]){
                count1++;
            }
        }
    }

    for(let i=0;i<nums2.length;i++){
        for(let j=0;j<newNum1.length;j++){
            if(nums2[i]==newNum1[j]){
                count2++;
            }
        }
    }

    let result=[count1,count2];
    console.log(result);

};
```

let nums1=[2,3,2];

let nums2=[1,2];

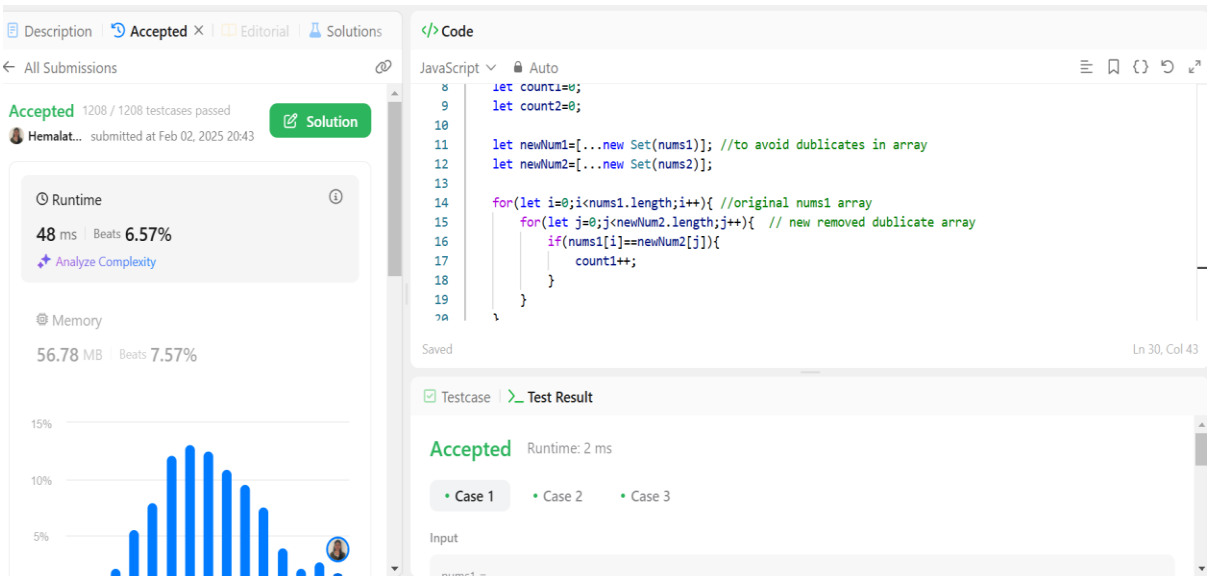
findIntersectionValues(nums1, nums2);

Output:

Leet-code submission link:

https://leetcode.com/problems/find-common-elements-between-two-arrays/submissions/1528758533/

Screenshot:



Conclusion:

Time complexity: O(n1.n2)

- n1 is the length of nums1 array and n2 is the length of nums2 array.
- For first nested loop takes O(n1*n2) complexity
- For second nested loop tales O(n2*n1) complexity.

Space complexity: O(k1+k2)

- k1 and k2 represent the unique elements in nums1 and nums2 array.
- Extra space required for new unique elements array

Problem 4:

Number of good pairs

Given an array of integers nums, return the number of good pairs.

A pair (i, j) is called good if nums[i] == nums[j] and i < j.

Example 1:

Input: nums = [1,2,3,1,1,3]

Output: 4

Question link:

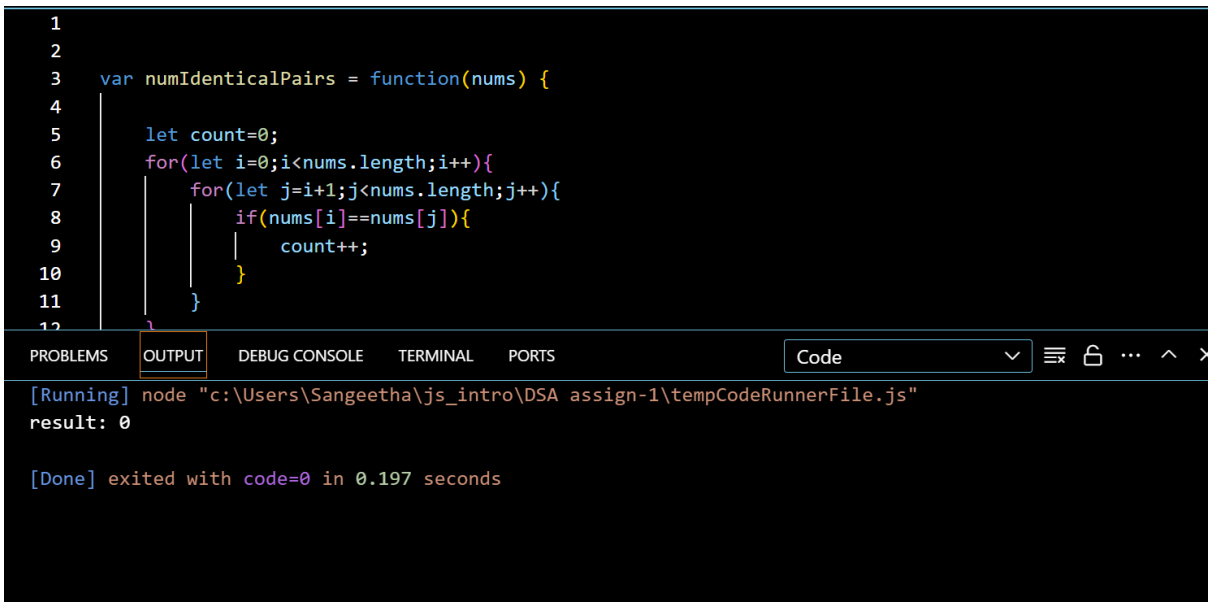
https://leetcode.com/problems/number-of-good-pairs/description/

Code:

```
var numIdenticalPairs = function(nums) {
    let count=0;
    for(let i=0;i<nums.length;i++){
        for(let j=i+1;j<nums.length;j++){
            if(nums[i]==nums[j]){
                count++;
            }
        }
    }
```

```
    console.log("result:",count);


};

let nums = [1,2,3];

numIdenticalPairs(nums);
```

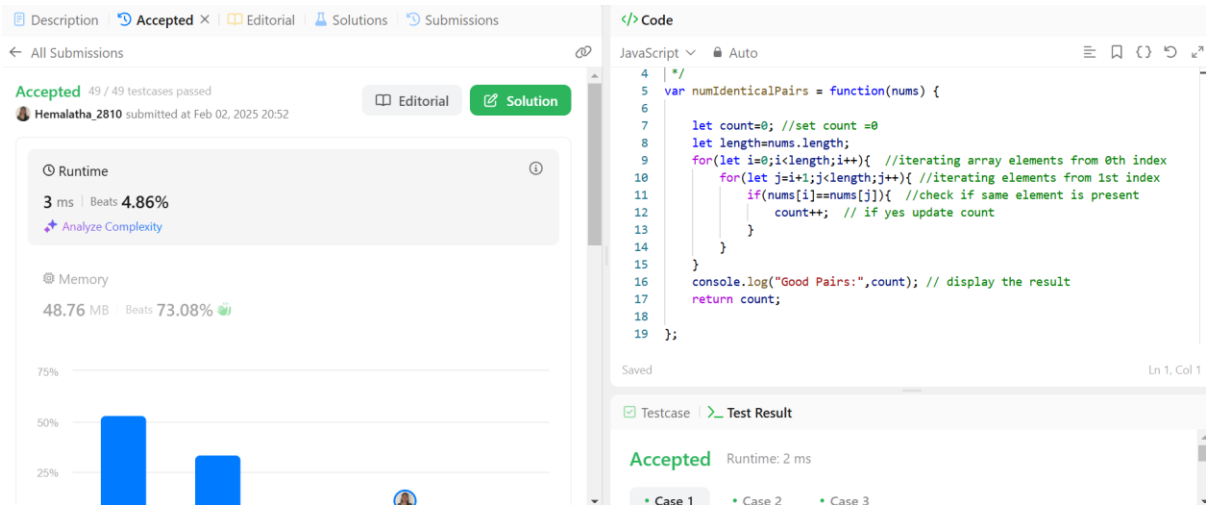
Output:



Leet-code submission link:

https://leetcode.com/problems/number-of-good-pairs/submissions/1528765846/

Screenshot:



Conclusion:

Time complexity: $O(n^2)$

Loop iteration = $O(n^2)$

Comparison= O(1)

Space complexity: O(1)

No extra space taken.

Problem-5:

Shuffle the array

Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn].

Return the array in the form [x1,y1,x2,y2,...,xn,yn].

Example:

Input: nums = [2,5,1,3,4,7], n = 3

Output: [2,3,5,4,1,7]

Question link:

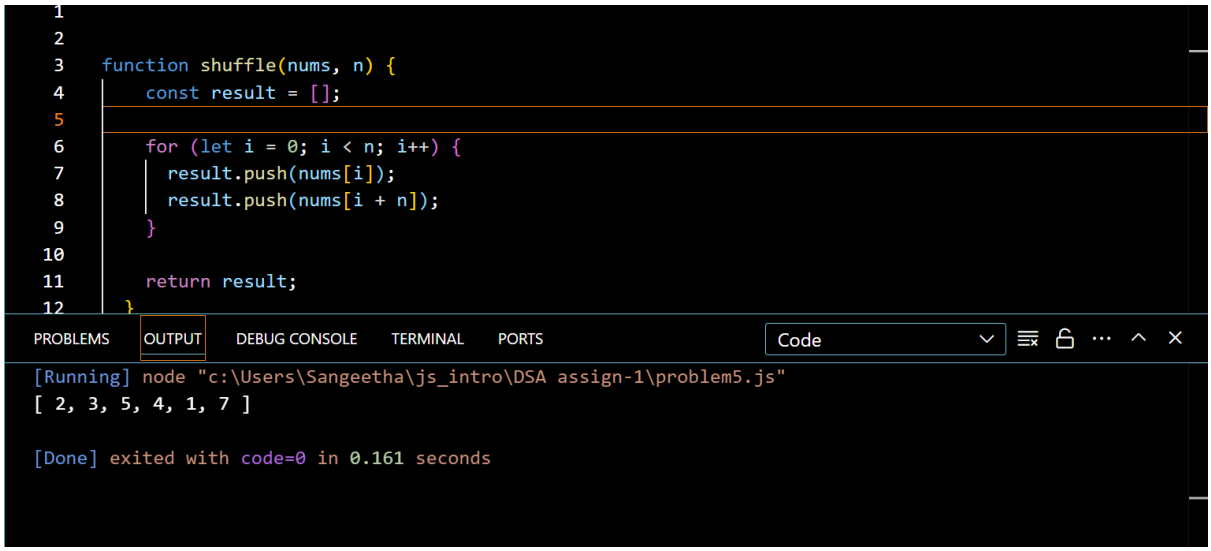
https://leetcode.com/problems/shuffle-the-array/description/

Code:

```
function shuffle(nums, n) {
    const result = [];
    for (let i = 0; i < n; i++) {
      result.push(nums[i]);
      result.push(nums[i + n]);
    }
    return result;
```

```
}

const nums = [2, 5, 1, 3, 4, 7];

const n = 3;

console.log(shuffle(nums, n));
```

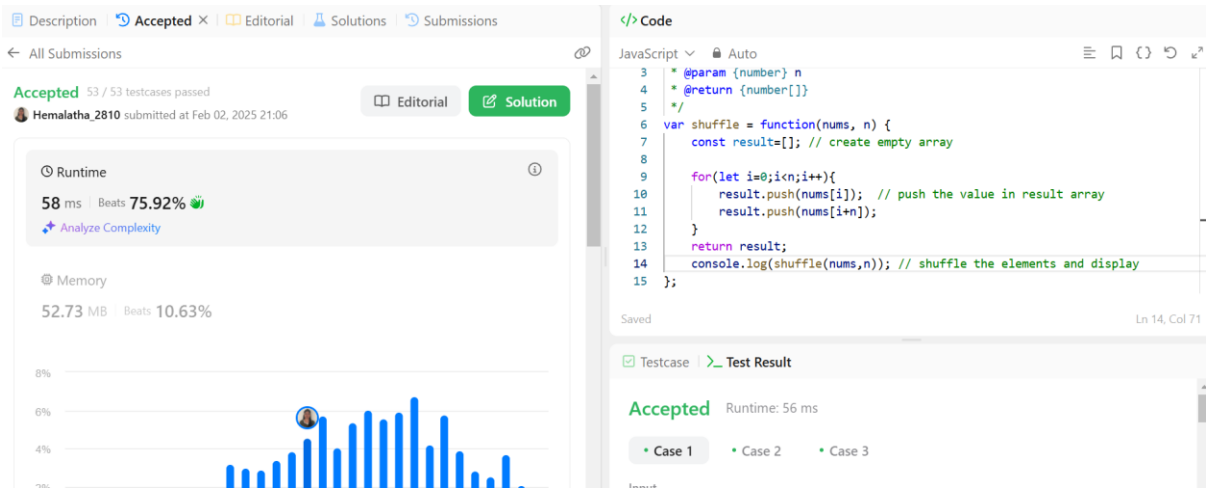
Output:



Leet-code submission link:

https://leetcode.com/problems/shuffle-the-array/submissions/1528778159/

Screen shot:



Conclusion:

Time complexity: O(n)

Loop and push operation-O(n)

Space complexity: O(n)

The space required for array elements O(2n) which simplify to O(n)