INTERNSHALA TRAININGS
Career Ki Guarantee

DATA STRUCTURE AND ALGORITHMS
ASSIGNMENT – II [SORTING ALGORITHMS]

Problem 1:

Two sum

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

Example :

Input: nums = [2,7,11,15], target = 9

Output: [0,1]

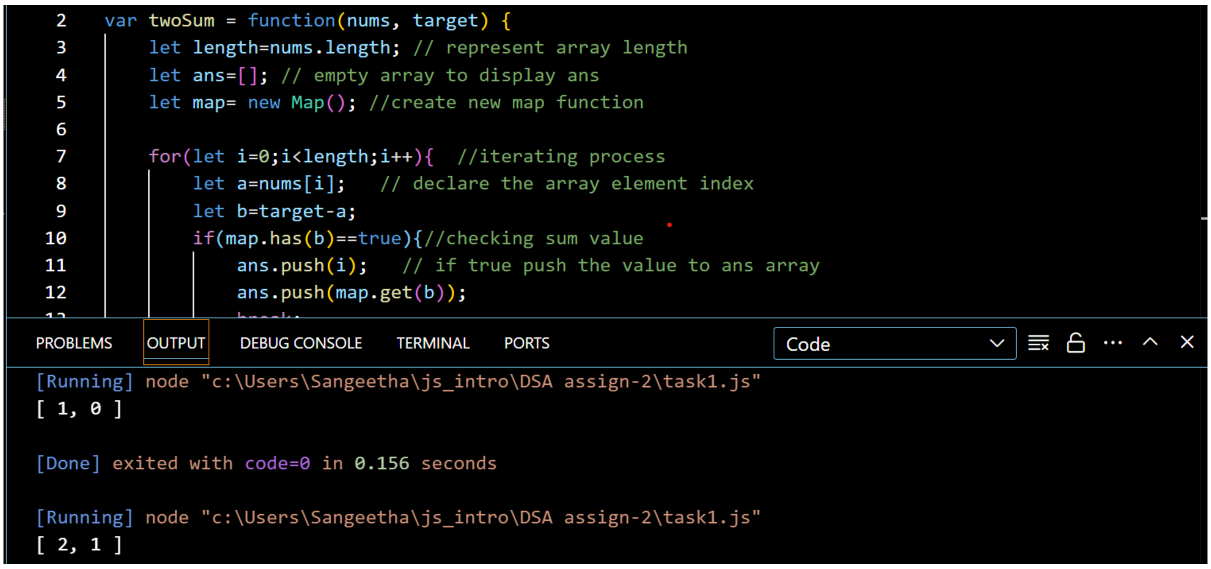
Question link:

https://leetcode.com/problems/two-sum/description/

Code:

```
var twoSum = function(nums, target) {
    let length=nums.length; // represent array length
    let ans=[]; // empty array to display ans
    let map= new Map(); //create new map function
    for(let i=0;i<length;i++){   //iterating process
        let a=nums[i];   // declare the array element index
        let b=target-a;
        if(map.has(b)==true){//checking sum value
            ans.push(i);   // if true push the value to ans array
            ans.push(map.get(b));
            break;
        }
        else{
            map.set(a,i);  // if false set into map
        }
    }
    return ans; // to return the answer

};

let nums=[3,2,4];
let target=6;
const result=twoSum(nums, target);
console.log(result);
```

Output:



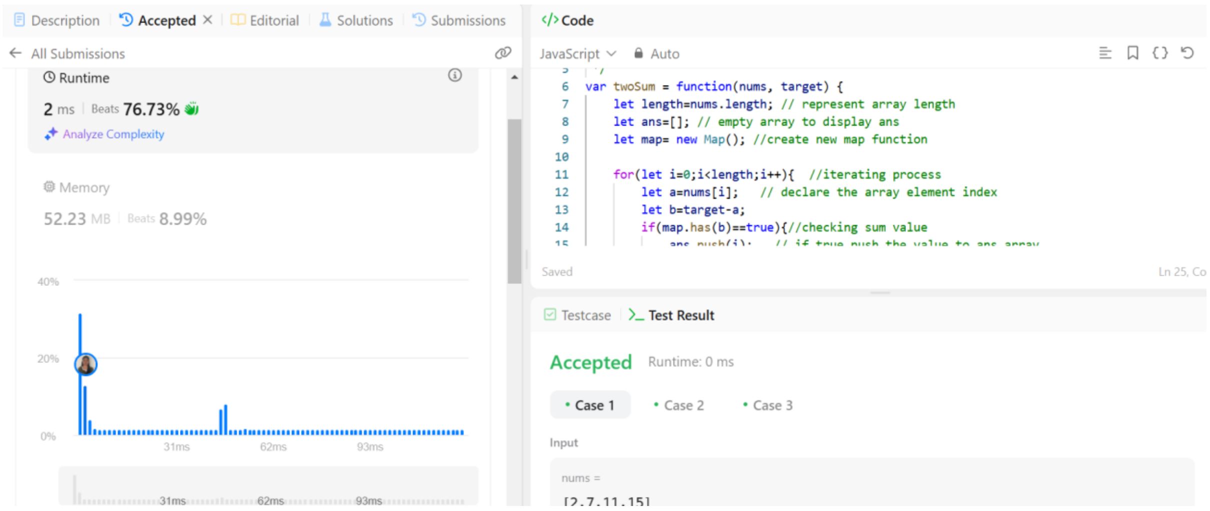
```
var twoSum = function(nums, target) {
    let length=nums.length; // represent array length
    let ans=[]; // empty array to display ans
    let map= new Map(); //create new map function

    for(let i=0;i<length;i++){  //iterating process
        let a=nums[i];   // declare the array element index
        let b=target-a;
        if(map.has(b)==true){//checking sum value
            ans.push(i);   // if true push the value to ans array
            ans.push(map.get(b));
```

```
[Running] node "c:\Users\Sangeetha\js_intro\DSA assign-2\task1.js"
[ 1, 0 ]

[Done] exited with code=0 in 0.156 seconds

[Running] node "c:\Users\Sangeetha\js_intro\DSA assign-2\task1.js"
[ 2, 1 ]
```

Leet-code submission link:

https://leetcode.com/problems/two-sum/submissions/1529746359/

Screenshot:



2 ms | Beats 76.73%

52.23 MB | Beats 8.99%

```
var twoSum = function(nums, target) {
    let length=nums.length; // represent array length
    let ans=[]; // empty array to display ans
    let map= new Map(); //create new map function

    for(let i=0;i<length;i++){  //iterating process
        let a=nums[i];   // declare the array element index
        let b=target-a;
        if(map.has(b)==true){//checking sum value
```

Accepted Runtime: 0 ms

Input

nums =
[2,7,11,15]

Conclusion:

Time complexity: O(n)

- Each element iterate once in nums array n= nums.length
- For map function ,it takes O(1)

Space complexity: O(n)

- The map will store all n elements, resulting in O(n) space usage. Ans array and other variables takes O(1) space.

Problem 2:

3Sum

Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.Notice that the solution set must not contain duplicate triplets.

Example:

Input: nums = [-1,0,1,2,-1,-4]

Output: [[-1,-1,2],[-1,0,1]]

Question link:

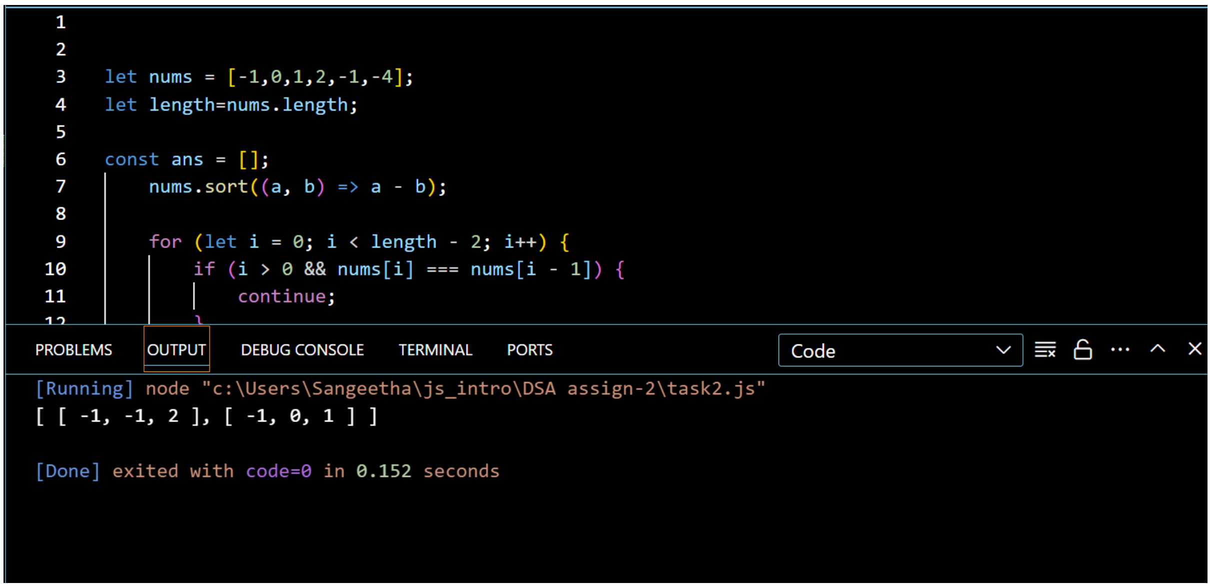
https://leetcode.com/problems/3sum/description/

Code:

```
let nums = [-1,0,1,2,-1,-4];

let length=nums.length;


const ans = [];

    nums.sort((a, b) => a - b);


    for (let i = 0; i < length - 2; i++) {

        if (i > 0 && nums[i] === nums[i - 1]) {

            continue;

        }
        let leftEle = i + 1;

        let rightEle = length - 1;



        while (leftEle < rightEle) {

            const sum = nums[i] + nums[leftEle] + nums[rightEle];



            if (sum === 0) {

                ans.push([nums[i], nums[leftEle], nums[rightEle]]);







                while (leftEle < rightEle && nums[leftEle] === nums[leftEle + 1]) {

                    leftEle++;

                }



                while (leftEle < rightEle && nums[rightEle] === nums[rightEle -
1]){

                    rightEle--;

                }



                leftEle++;

                rightEle--;

            } else if (sum < 0) {
```

```
            leftEle++;
        } else {
            rightEle--;
        }
    }
}

console.log(ans);
```

Output:



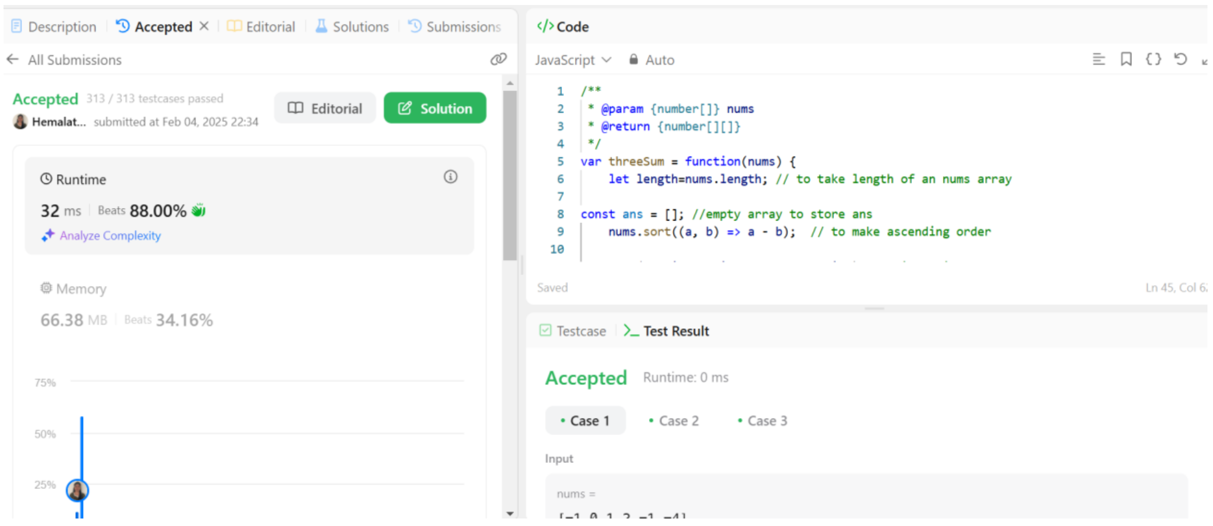
```
let nums = [-1,0,1,2,-1,-4];
let length=nums.length;

const ans = [];
    nums.sort((a, b) => a - b);

    for (let i = 0; i < length - 2; i++) {
        if (i > 0 && nums[i] === nums[i - 1]) {
            continue;
```

```
[Running] node "c:\Users\Sangeetha\js_intro\DSA assign-2\task2.js"
[ [ -1, -1, 2 ], [ -1, 0, 1 ] ]

[Done] exited with code=0 in 0.152 seconds
```

Leet-code submission link:

https://leetcode.com/problems/3sum/submissions/1531297396/

Screenshot:



Accepted 313 / 313 testcases passed

Hemalat... submitted at Feb 04, 2025 22:34

Runtime

32 ms | Beats 88.00%

Memory

66.38 MB | Beats 34.16%

```javascript
/**
 * @param {number[]} nums
 * @return {number[][]}
 */
var threeSum = function(nums) {
    let length=nums.length; // to take length of an nums array

const ans = []; //empty array to store ans
    nums.sort((a, b) => a - b);  // to make ascending order
```

Saved

Ln 45, Col 6

Accepted Runtime: 0 ms

Input

nums =

Conclusion:

Time complexity O(n^2)

- For sorting the array takes O(n log n)
- The outer and inner loops takes O(n^2)
- So, O(n log n)+ O(n^2)= O(n^2)

Space complexity O(k)

- K means output ans array and sorting takes O(1) space required to run.

Problem-3:

Long pressed name

Your friend is typing his name into a keyboard. Sometimes, when typing a character c, the key might get long pressed, and the character will be typed 1 or more times.You examine the typed characters of the keyboard. Return True if it is possible that it was your friends name, with some characters (possibly none) being long pressed.

Example 1:

Input: name = "alex", typed = "aaleex"

Output: true

Question link:

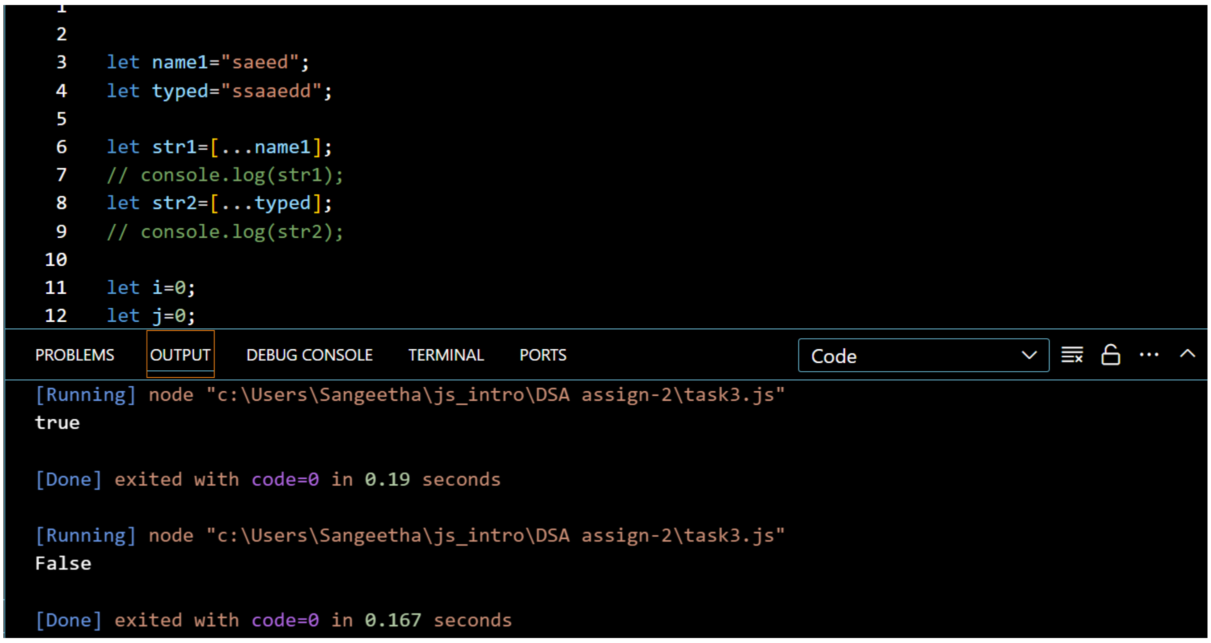
https://leetcode.com/problems/long-pressed-name/description/

Code:

```
let name1="saeed";

let typed="ssaaedd";

let str1=[...name1];

// console.log(str1);

let str2=[...typed];

// console.log(str2);


let i=0;

let j=0;
```

```
while(i<str1.length){
    if(str1[i]===str2[j]){
          i++;
          j++;
    }else if(str2[j]==str2[j-1]){
        j++;
    }else{
        console.log("False");
        return false;
    }


    }
    console.log("true");
```

Output:



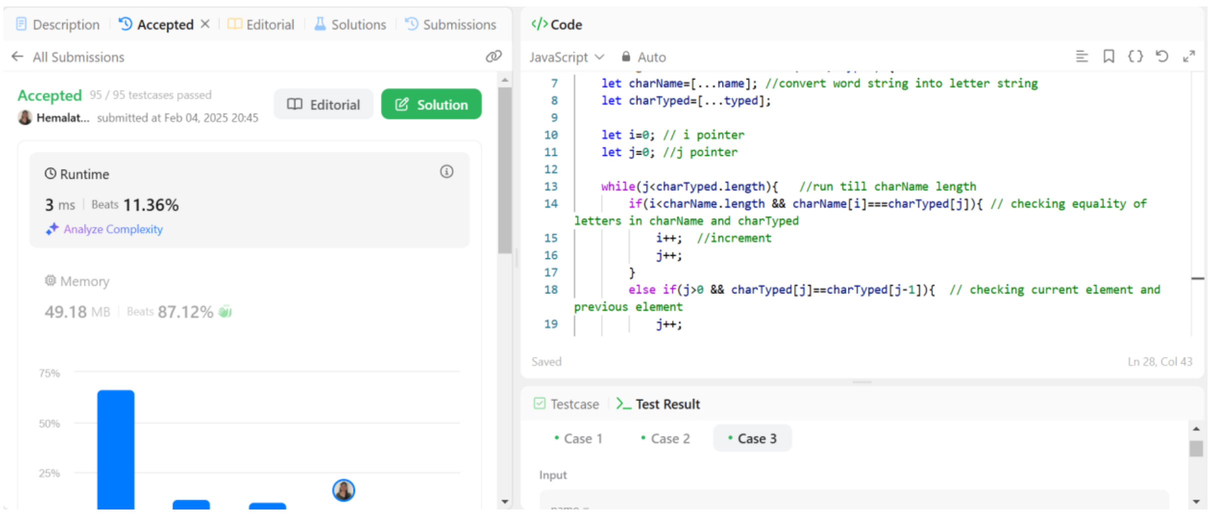
```
let name1="saeed";
let typed="ssaaedd";

let str1=[...name1];
// console.log(str1);
let str2=[...typed];
// console.log(str2);

let i=0;
let j=0;
```

```
[Running] node "c:\Users\Sangeetha\js_intro\DSA assign-2\task3.js"
true

[Done] exited with code=0 in 0.19 seconds

[Running] node "c:\Users\Sangeetha\js_intro\DSA assign-2\task3.js"
False

[Done] exited with code=0 in 0.167 seconds
```

Leet-code submission link:

https://leetcode.com/problems/long-pressed-name/submissions/1531164828/

Screen shot:



Accepted 95 / 95 testcases passed

Hemalat... submitted at Feb 04, 2025 20:45

Runtime: 3 ms | Beats 11.36%

Memory: 49.18 MB | Beats 87.12%

```
    let charName=[...name]; //convert word string into letter string
    let charTyped=[...typed];

    let i=0; // i pointer
    let j=0; //j pointer

    while(j<charTyped.length){   //run till charName length
        if(i<charName.length && charName[i]===charTyped[j]){ // checking equality of
letters in charName and charTyped
            i++;  //increment
            j++;
        }
        else if(j>0 && charTyped[j]==charTyped[j-1]){  // checking current element and
previous element
            j++;
```

Saved

Ln 28, Col 43

Conclusion:

Time complexity : O(n)

The loop runs till length of an array

Space complexity: O(1)

No extra space required

Problem 4:

Make chunks to make sorted

You are given an integer array arr of length n that represents a permutation of the integers in the range [0, n - 1].

We split arr into some number of chunks (i.e., partitions), and individually sort each chunk. After concatenating them, the result should equal the sorted array.Return the largest number of chunks we can make to sort the array.

Example:

Input: arr = [4,3,2,1,0]

Output: 1

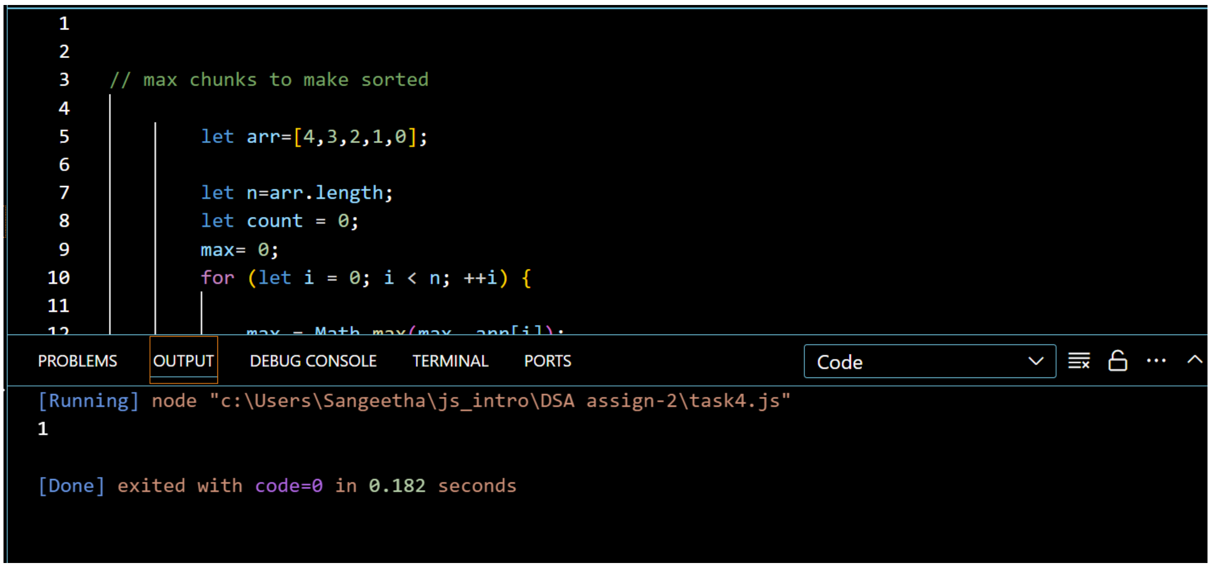
Question link:

https://leetcode.com/problems/max-chunks-to-make-sorted/description/

Code:

```
let arr=[4,3,2,1,0];


    let n=arr.length;

    let count = 0;

    max= 0;

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


        max = Math.max(max, arr[i]);
        if (max == i)
            count++;
    }

console.log(count);
```

Output:



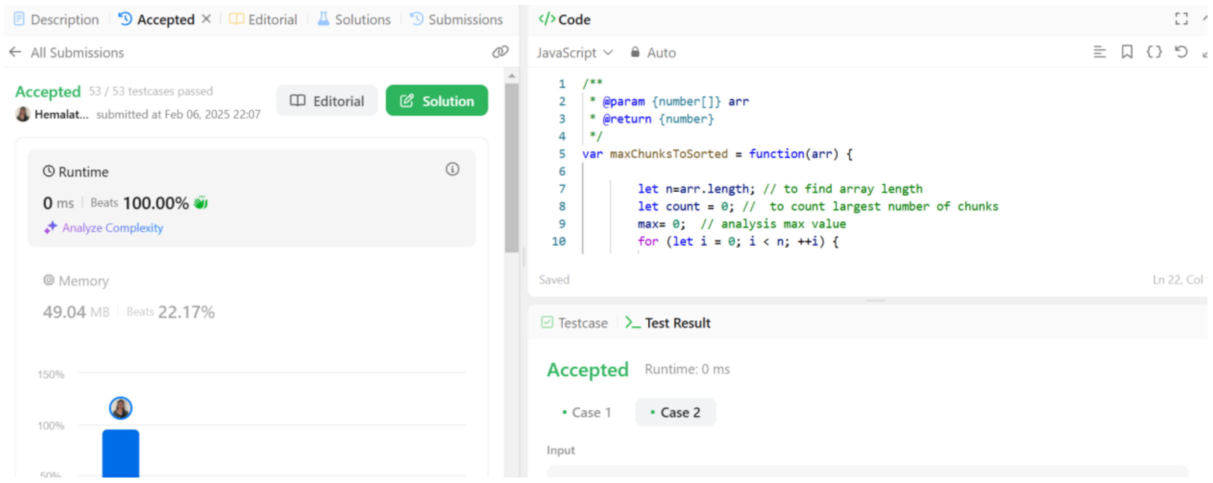
```
// max chunks to make sorted

        let arr=[4,3,2,1,0];

        let n=arr.length;
        let count = 0;
        max= 0;
        for (let i = 0; i < n; ++i) {

            max = Math.max(max, arr[i]);
```

```
[Running] node "c:\Users\Sangeetha\js_intro\DSA assign-2\task4.js"
1

[Done] exited with code=0 in 0.182 seconds
```

Code submission link:

https://leetcode.com/problems/max-chunks-to-make-sorted/submissions/1533788699/

Screenshot:



Accepted 53 / 53 testcases passed

Hemalat... submitted at Feb 06, 2025 22:07

Runtime

0 ms | Beats 100.00%

Memory

49.04 MB | Beats 22.17%

```
/**
 * @param {number[]} arr
 * @return {number}
 */
var maxChunksToSorted = function(arr) {

        let n=arr.length; // to find array length
        let count = 0; //  to count largest number of chunks
        max= 0;  // analysis max value
        for (let i = 0; i < n; ++i) {
```

Saved | Ln 22, Col

Accepted Runtime: 0 ms

Conclusion

Time complexity : O(n)

The for loop runs till length of an given array

Space complexity : O(1)

No extra space required for code

Problem-5

Sort colours

Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue.We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.You must solve this problem without using the library's sort function.

Example

Input: nums = [2,0,2,1,1,0]

Output: [0,0,1,1,2,2]

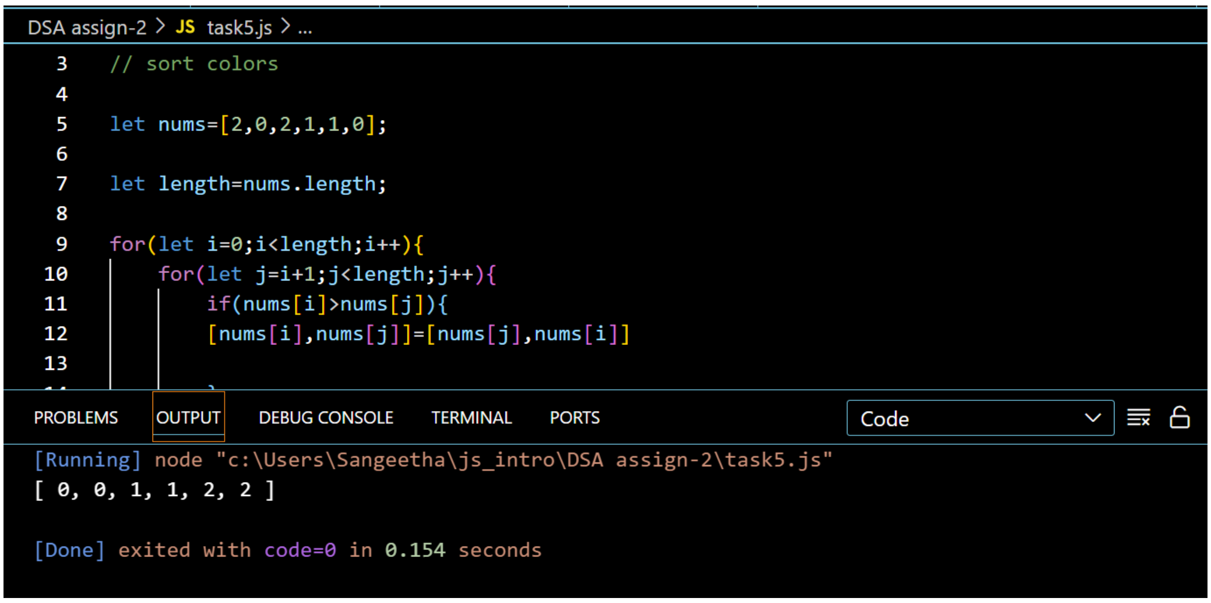
Question link

https://leetcode.com/problems/sort-colors/description/

Code

```
let nums=[2,0,2,1,1,0];

let length=nums.length;

for(let i=0;i<length;i++){
    for(let j=i+1;j<length;j++){
        if(nums[i]>nums[j]){
        [nums[i],nums[j]]=[nums[j],nums[i]]

        }
    }
}
console.log(nums);
```

Output



DSA assign-2 > JS task5.js > ...

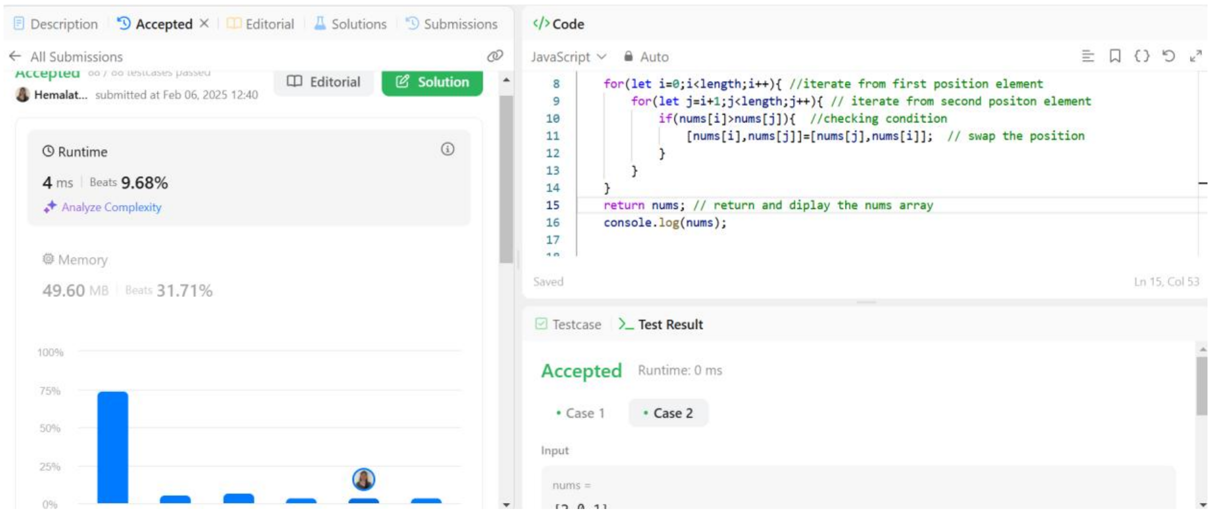
```
// sort colors

let nums=[2,0,2,1,1,0];

let length=nums.length;

for(let i=0;i<length;i++){
    for(let j=i+1;j<length;j++){
        if(nums[i]>nums[j]){
        [nums[i],nums[j]]=[nums[j],nums[i]]
```

```
[Running] node "c:\Users\Sangeetha\js_intro\DSA assign-2\task5.js"
[ 0, 0, 1, 1, 2, 2 ]

[Done] exited with code=0 in 0.154 seconds
```

Code submission link:

https://leetcode.com/problems/sort-colors/submissions/1533286845/

Screenshot:



Hemalat... submitted at Feb 06, 2025 12:40

Runtime
4 ms | Beats 9.68%

Memory
49.60 MB | Beats 31.71%

```
    for(let i=0;i<length;i++){ //iterate from first position element
        for(let j=i+1;j<length;j++){ // iterate from second positon element
            if(nums[i]>nums[j]){  //checking condition
                [nums[i],nums[j]]=[nums[j],nums[i]];  // swap the position
            }
        }
    }
    return nums; // return and diplay the nums array
    console.log(nums);
```

Saved   Ln 15, Col 53

Accepted Runtime: 0 ms

Input

nums =

Conclusion

Time complexity : $O(n^2)$

- The outer loop runs n times (length of an array) and inner loop runs (n-1),(n-2)…

  $n(n-1)/2 \quad = O(n^2)$

Space complexity : $O(1)$

- No extra space required

Problem 6:

Maximum sub-array

Given an integer array nums, find the subarray with the largest sum, and return its sum.

Example:

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

Output: 6

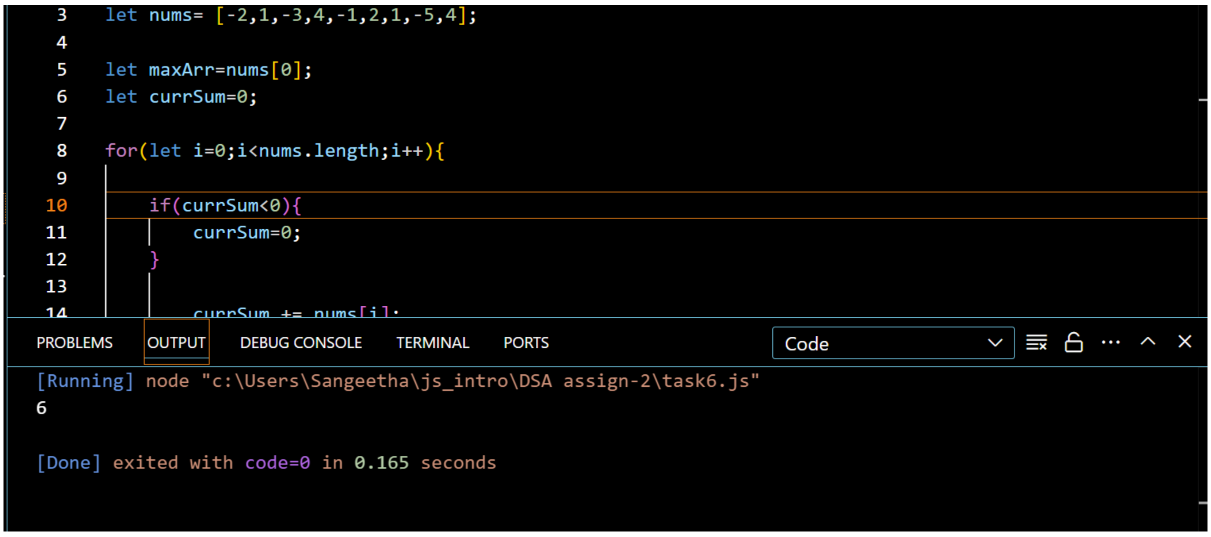
Question link:

https://leetcode.com/problems/maximum-subarray/description/

Code:

```
let nums= [-2,1,-3,4,-1,2,1,-5,4];


let maxArr=nums[0];

let currSum=0;


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


    if(currSum<0){

        currSum=0;

    }


        currSum += nums[i];

        maxArr=Math.max(maxArr , currSum);
}
console.log(maxArr);
```

Output:



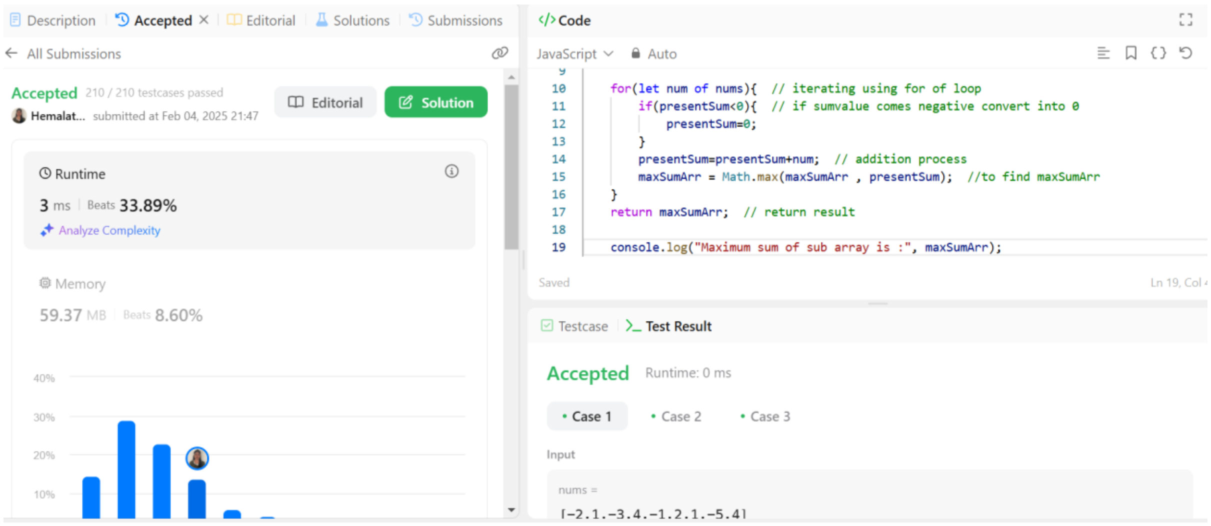
```
let nums= [-2,1,-3,4,-1,2,1,-5,4];

let maxArr=nums[0];
let currSum=0;

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

    if(currSum<0){
        currSum=0;
    }

        currSum += nums[i];
```

```
[Running] node "c:\Users\Sangeetha\js_intro\DSA assign-2\task6.js"
6

[Done] exited with code=0 in 0.165 seconds
```

Leet-code submission link:

https://leetcode.com/problems/maximum-subarray/submissions/1531236376/

Screenshot:



Accepted 210 / 210 testcases passed
Hemalat... submitted at Feb 04, 2025 21:47

Runtime
3 ms | Beats 33.89%

Memory
59.37 MB | Beats 8.60%

```
    for(let num of nums){  // iterating using for of loop
        if(presentSum<0){  // if sumvalue comes negative convert into 0
            presentSum=0;
        }
        presentSum=presentSum+num;  // addition process
        maxSumArr = Math.max(maxSumArr , presentSum);  //to find maxSumArr
    }
    return maxSumArr;  // return result

    console.log("Maximum sum of sub array is :", maxSumArr);
```

Saved   Ln 19, Col

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

Input

nums =
[-2,1,-3,4,-1,2,1,-5,4]

Conclusion:

Time complexity: O(n)

for loop is iterating till length of an nums array.

Space complexity: O(1)

No extra space is required .

Problem 7

Product of array except self

Given an integer array nums, return an array answer such that answer[i] is equal to the product of all the elements of nums except nums[i].The product of any prefix or suffix of nums is guaranteed to fit in a 32-bit integer.You must write an algorithm that runs in O(n) time and without using the division operation.

Example

Input: nums = [1,2,3,4]

Output: [24,12,8,6]

Question link

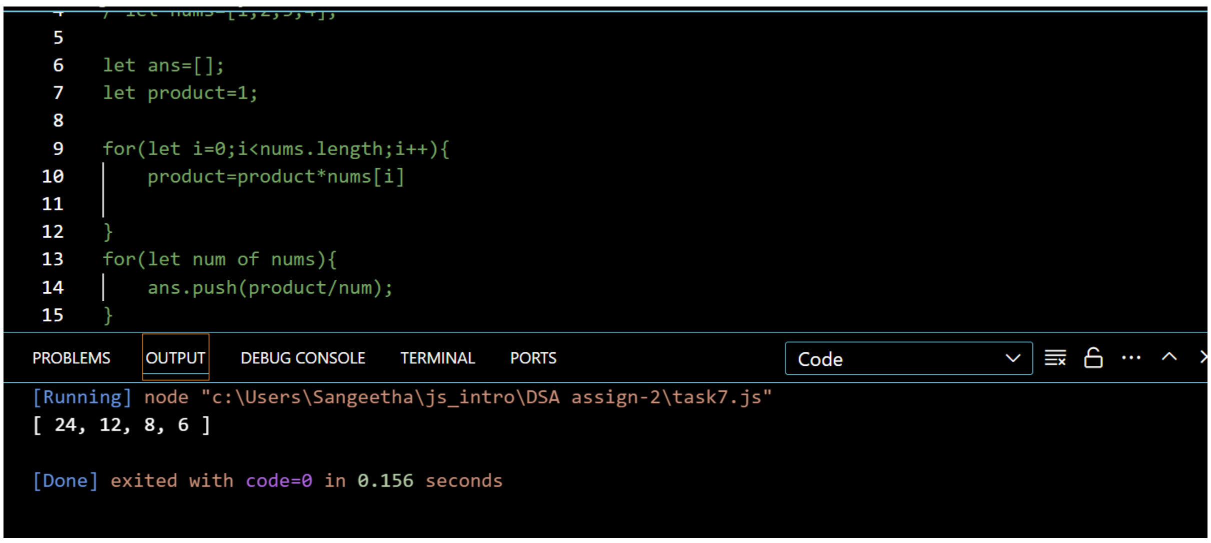
https://leetcode.com/problems/product-of-array-except-self/description/

Code

```
let nums=[1,2,3,4];

let ans=[];
let length=nums.length;

let leftProduct = 1;
for(let i=0;i<length;i++){
    ans[i] = leftProduct;
    leftProduct*=nums[i];

}
```

```
let rightProduct = 1;
for(let i=length-1;i>=0;i--){
    ans[i]*=rightProduct;
    rightProduct*=nums[i];

}

console.log(ans);
```

Output



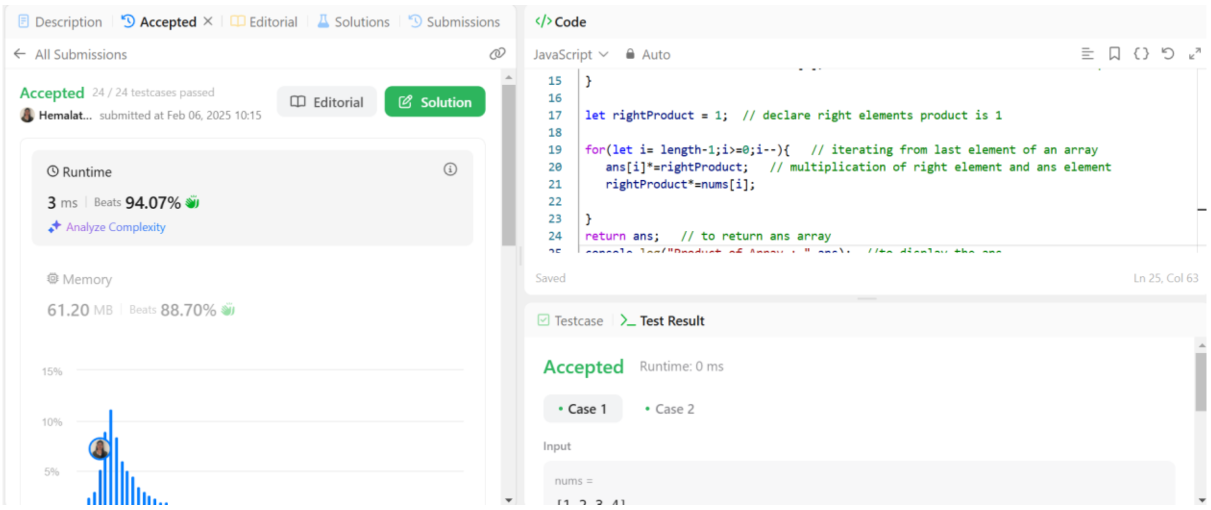
```
// let nums=[1,2,3,4];

let ans=[];
let product=1;

for(let i=0;i<nums.length;i++){
    product=product*nums[i]

}
for(let num of nums){
    ans.push(product/num);
}
```

```
[Running] node "c:\Users\Sangeetha\js_intro\DSA assign-2\task7.js"
[ 24, 12, 8, 6 ]

[Done] exited with code=0 in 0.156 seconds
```

Code submission link:

https://leetcode.com/problems/product-of-array-except-self/submissions/1533141303/

Screenshot:



```
}

let rightProduct = 1;  // declare right elements product is 1

for(let i= length-1;i>=0;i--){   // iterating from last element of an array
   ans[i]*=rightProduct;   // multiplication of right element and ans element
   rightProduct*=nums[i];

}
return ans;   // to return ans array
```

Conclusion:

Time Complexity: O(n)

The loop runs till length(n) of an array

Space complexity: O(n)

The algorithm uses extra array ans=[ ].