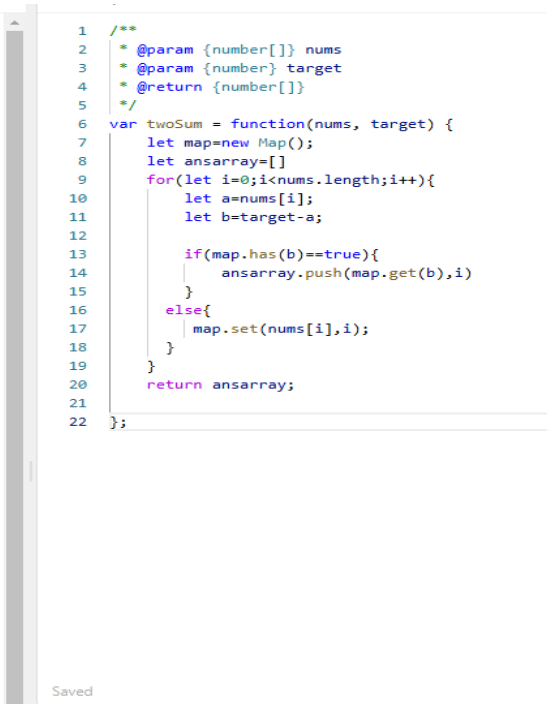
1) Question: https://leetcode.com/problems/two-sum/description/

Solution Link:- https://leetcode.com/problems/two-sum/submissions/1485546755

Time Complexity: O(n)

Space Complexity: O(n)

Screenshot:



Description:

Time complexity:O(n)

Iterates all over the n sized array and there is only one loop and pushes the index to the ansarray

Space complexity: O(n)

The result array keep on change with respect to the size of the array and using the map data structure

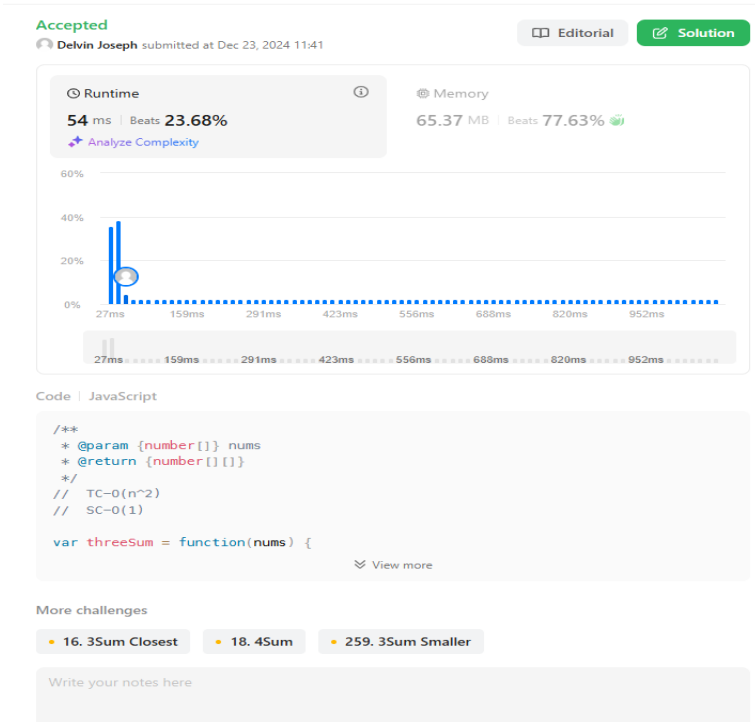
2) Question: https://leetcode.com/problems/3sum/description/

Solution Link: https://leetcode.com/problems/3sum/submissions/1486013751

Time Complexity: O(n^2)

Space Complexity: O(1)

Screenshot:



```
// TC-O(n^2)
// SC-O(1)
var threeSum = function(nums) {
  const result=[];

    //selection sort

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

    //     for(let j=i+1;j<nums.length;j++){
    //         if(nums[j]<nums[min_index]){
    //             min_index=j;
    //         }
    //     }
    //         let temp=nums[min_index];
    //         nums[min_index]=nums[i];
    //         nums[i]=temp;

    // }
    nums.sort((a,b)=>a-b);
    for(let i=0;i<nums.length-2;i++){
        if(i>0 && nums[i]==nums[i-1]) continue;
        let left=i+1;
        let right= nums.length-1;

        while(left<right){
            const sum=nums[i]+nums[left]+nums[right];

            if(sum==0){
                result.push([nums[i],nums[left],nums[right]]);

                while(left<right && nums[left]==nums[left+1]) left++;
                while(left<right && nums[right]==nums[right-1]) right--;
                left++;
                right++;
            }
            else if(sum<0){
                left++;
            }else{
                right--;
            }
        }
    }
    return result;
};
```

Description:

Time complexity:O(n^2)

Iterates all over the n sized array which is outer loop for the inner loop also it iterates to n sized array so TC- O(n^2)

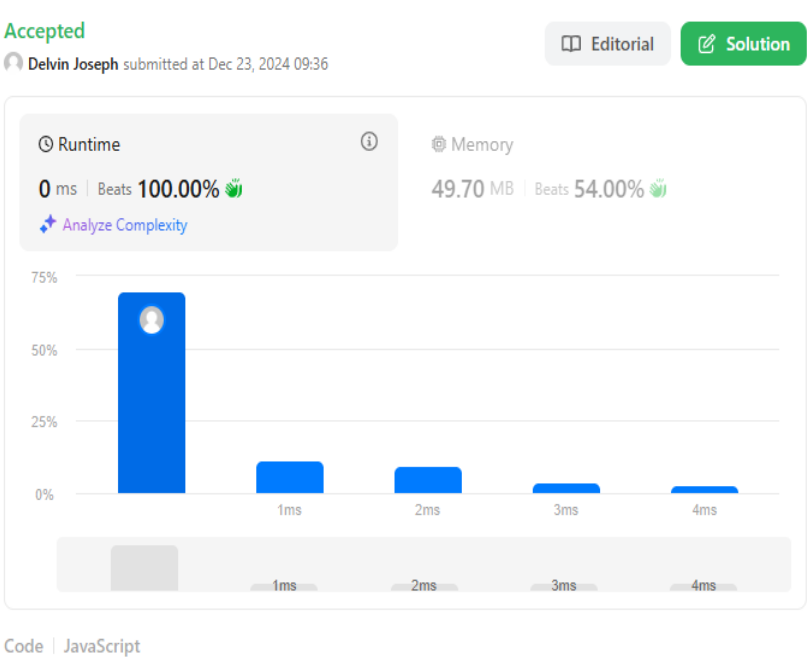
Space complexity: O(1)

No extra space is taken it constant sapce

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

Solution: https://leetcode.com/problems/long-pressed-name/submissions/1485933341/

Screenshot:



Time Complexity: O(n)

Space Complexity: O(1)

Description:

Time Complexity: O(n)

It iterates to n sized array so the TC is O(n) the n is arr.length

Space Complexity: O(1)

No extra space is taking the space is constant which is O(1)

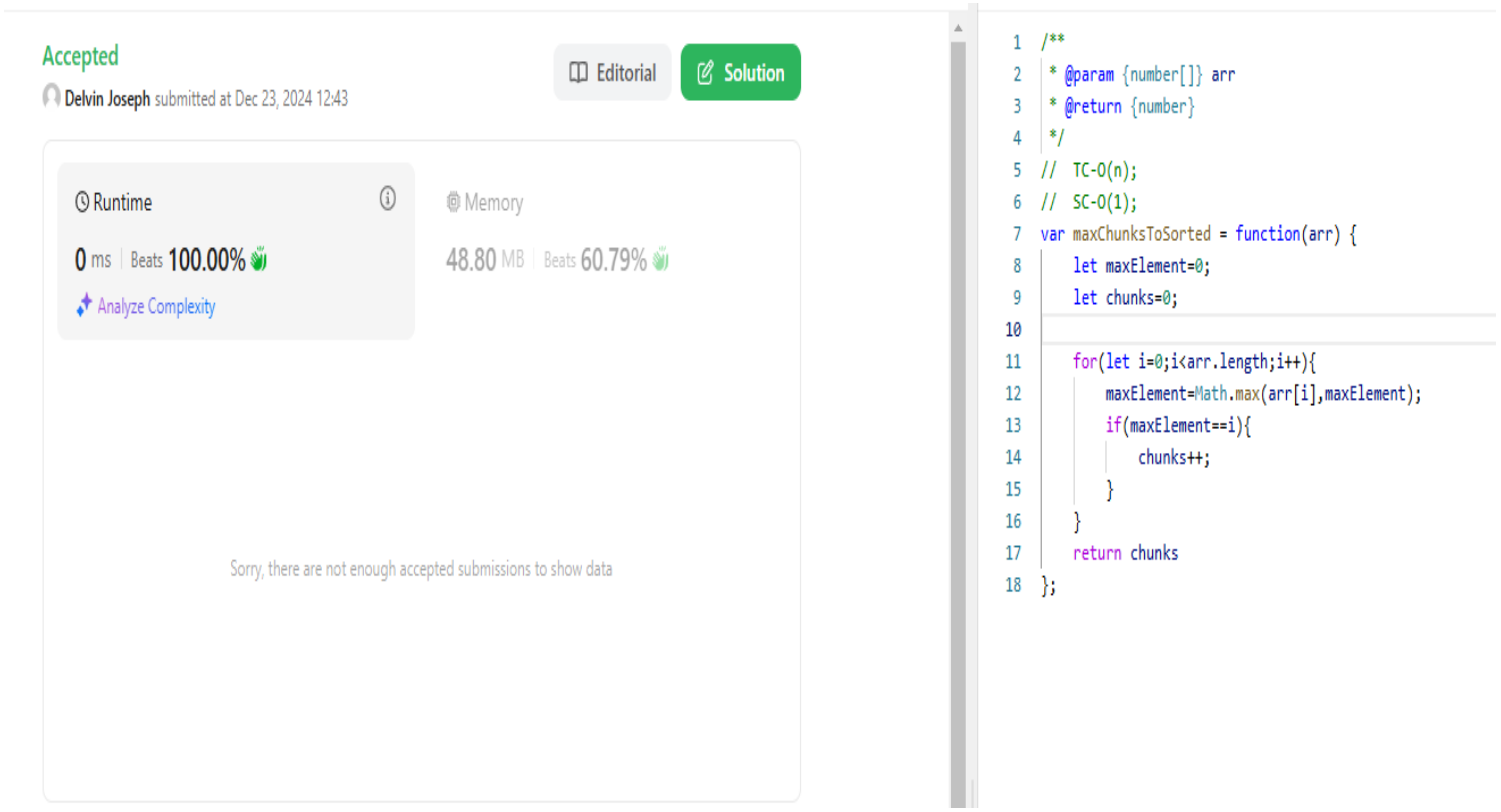
4)Question: https://leetcode.com/problems/max-chunks-to-make-sorted/description/

Solution: https://leetcode.com/problems/max-chunks-to-make-sorted/submissions/1486060833

Time Complexity: O(n)

Space Complexity: O(1)

Screenshot:



Description:

Time Complexity:O(n)

It iterates all over the n –sized array there for the time complexity is O(n)

Space complexity is O(1) because there is no extra space is taking it is constant

5)Question: https://leetcode.com/problems/sort-colors/description/

Solution: https://leetcode.com/problems/sort-colors/submissions/1486121743

Time Complexity: O(n)

Space Complexity: O(1)

Screenshot:


```javascript
/**
 * @param {number[]} nums
 * @return {void} Do not return anything, modify nums in-place instead.
 */
// TC-O(n)
// SC-O(1)
var sortColors = function(nums) {
    for(let i=1;i<nums.length;i++){
        let key=nums[i];
        let j=i-1;

        while(j>=0 && nums[j]>key){
            nums[j+1]=nums[j];
            j--;
        }
        nums[j+1]=key;
    }
    return nums;
};
```

Description:

Time Complexity:O(n)

It iterates all over the n –sized array there for the time complexity is O(n)

Space complexity is O(1) because no extra space is taking is constant space

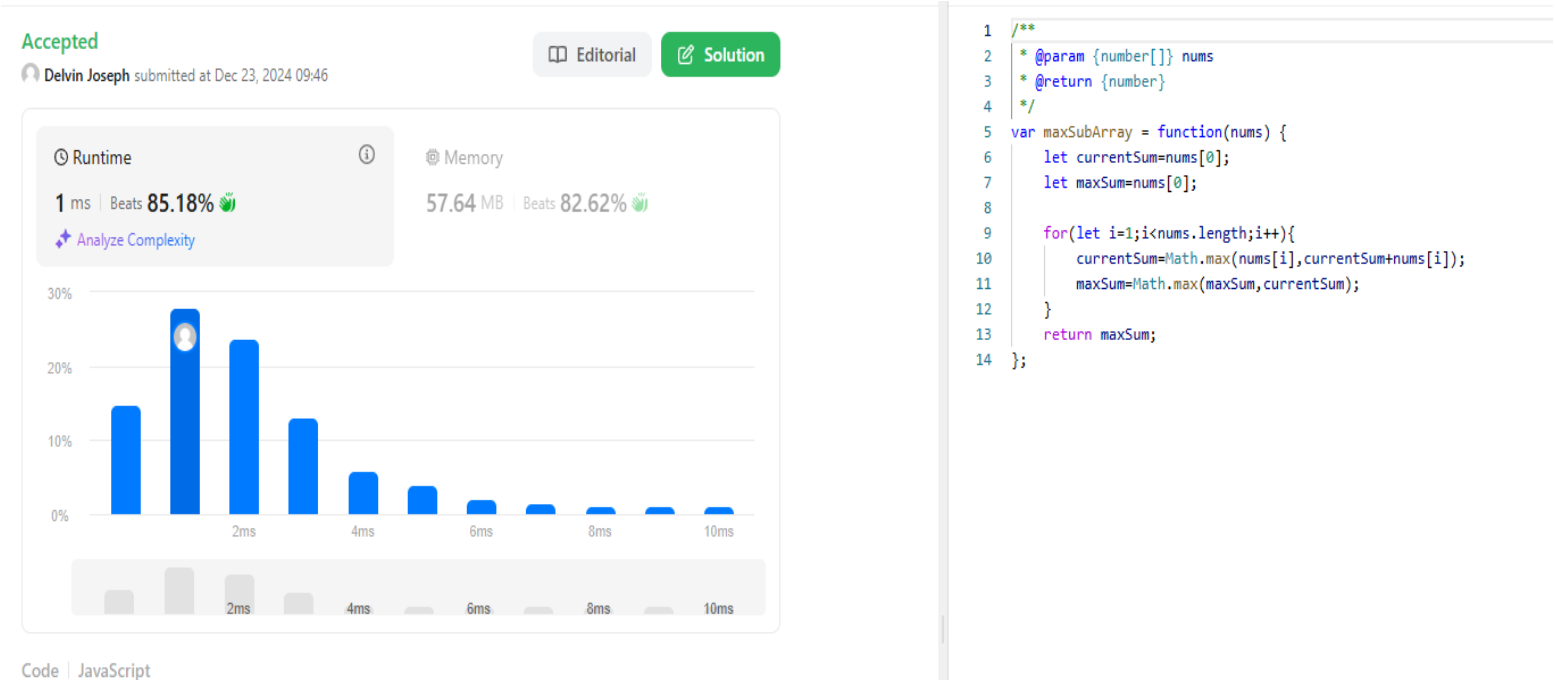
6)question: https://leetcode.com/problems/maximum-subarray/description/

Solution: https://leetcode.com/problems/maximum-subarray/submissions/1485938418

Time Complexity: O(n)

Space Complexity: O(1)

Screenshot:



Description:

Time Complexity:O(n)

It iterates all over the n –sized array there for the time complexity is O(n) where n is array length

Space complexity is O(1) because no extra space is taking is constant space

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

Solution: https://leetcode.com/problems/product-of-array-except-self/submissions/1486128459

Time Complexity: O(n)

Space Complexity: O(n)

Screenshot:

```javascript
/**
 * @param {number[]} nums
 * @return {number[]}
 */
// TC:O(n)
// SC:O(n)
var productExceptSelf = function(nums) {
    let n=nums.length;
    let answer=new Array(n).fill(1);

    let prefix=1;
    for(let i=0;i<n;i++){
        answer[i]=prefix;
        prefix*=nums[i];
    }

    let suffix=1;
    for(let i=n-1;i>=0;i--){
        answer[i]*=suffix;
        suffix*=nums[i]
    }
    return answer;
};
```

Description:

Time Complexity:(n)

It iterates all over the n –sized array there for the time complexity is O(n) where n is array length

Space complexity is O(n) because the space increase for the answer array when the array also increase so the SC is O(n)