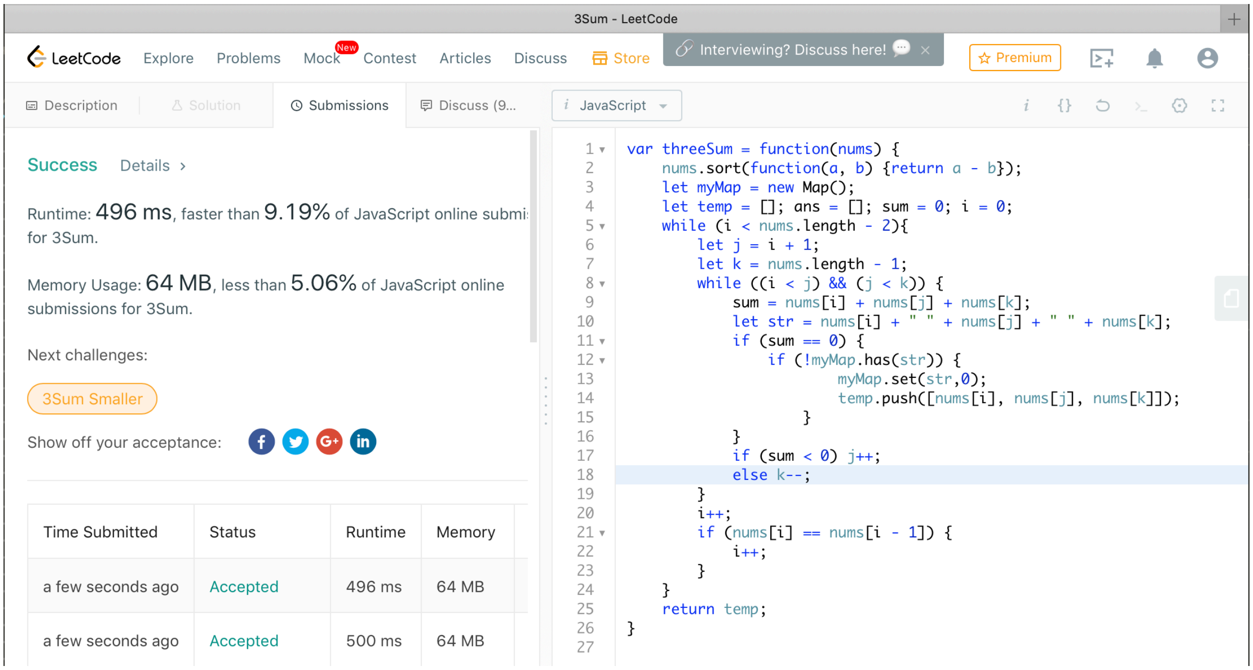
3Sum



Success Details ›

Runtime: 496 ms, faster than 9.19% of JavaScript online submi
for 3Sum.

Memory Usage: 64 MB, less than 5.06% of JavaScript online submissions for 3Sum.

Next challenges:

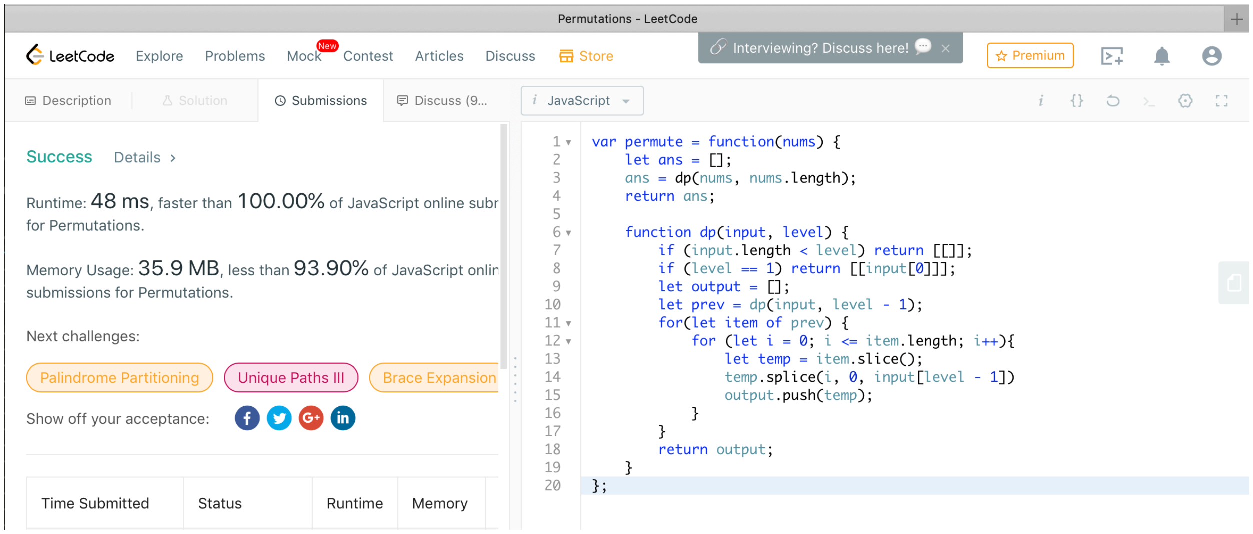
3Sum Smaller

Show off your acceptance:

Time Submitted	Status	Runtime	Memory
a few seconds ago	Accepted	496 ms	64 MB
a few seconds ago	Accepted	500 ms	64 MB

```javascript
var threeSum = function(nums) {
    nums.sort(function(a, b) {return a - b});
    let myMap = new Map();
    let temp = []; ans = []; sum = 0; i = 0;
    while (i < nums.length - 2){
        let j = i + 1;
        let k = nums.length - 1;
        while ((i < j) && (j < k)) {
            sum = nums[i] + nums[j] + nums[k];
            let str = nums[i] + " " + nums[j] + " " + nums[k];
            if (sum == 0) {
                if (!myMap.has(str)) {
                        myMap.set(str,0);
                        temp.push([nums[i], nums[j], nums[k]]);
                    }
            }
            if (sum < 0) j++;
            else k--;
        }
        i++;
        if (nums[i] == nums[i - 1]) {
            i++;
        }
    }
    return temp;
}
```

Permutations



Success Details ›

Runtime: 48 ms, faster than 100.00% of JavaScript online subr
for Permutations.

Memory Usage: 35.9 MB, less than 93.90% of JavaScript onlin
submissions for Permutations.

Next challenges:

Palindrome Partitioning Unique Paths III Brace Expansion

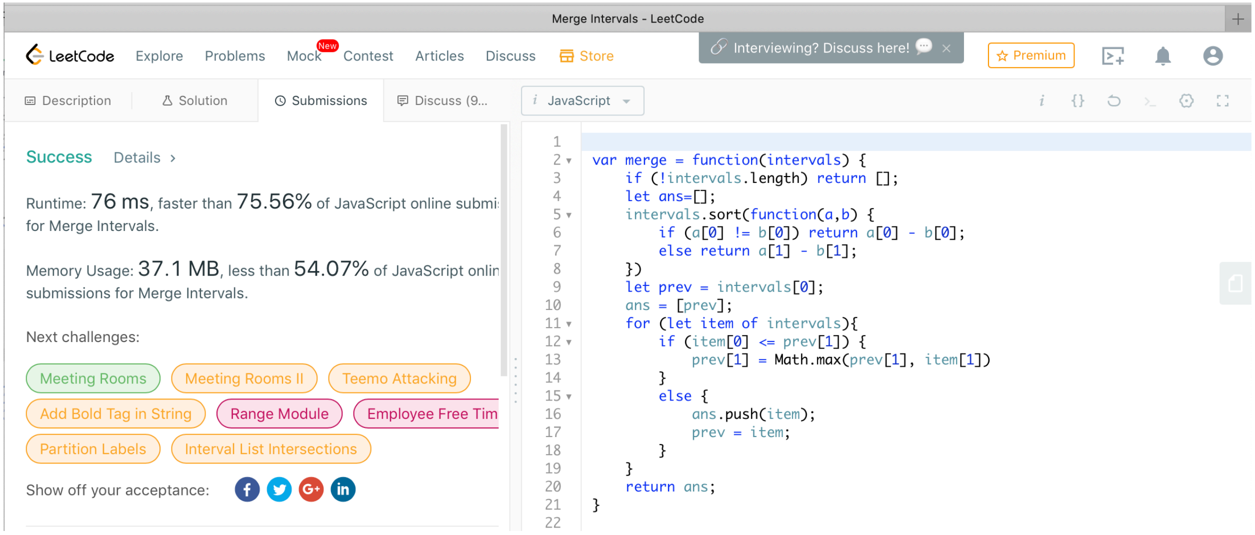
Show off your acceptance:

Time Submitted	Status	Runtime	Memory

```javascript
var permute = function(nums) {
    let ans = [];
    ans = dp(nums, nums.length);
    return ans;

    function dp(input, level) {
        if (input.length < level) return [[]];
        if (level == 1) return [[input[0]]];
        let output = [];
        let prev = dp(input, level - 1);
        for(let item of prev) {
            for (let i = 0; i <= item.length; i++){
                let temp = item.slice();
                temp.splice(i, 0, input[level - 1])
                output.push(temp);
            }
        }
        return output;
    }
};
```

Merge Intevals



Success Details ›

Runtime: 76 ms, faster than 75.56% of JavaScript online submi for Merge Intervals.

Memory Usage: 37.1 MB, less than 54.07% of JavaScript onlin submissions for Merge Intervals.

Next challenges:

Meeting Rooms | Meeting Rooms II | Teemo Attacking | Add Bold Tag in String | Range Module | Employee Free Tim | Partition Labels | Interval List Intersections

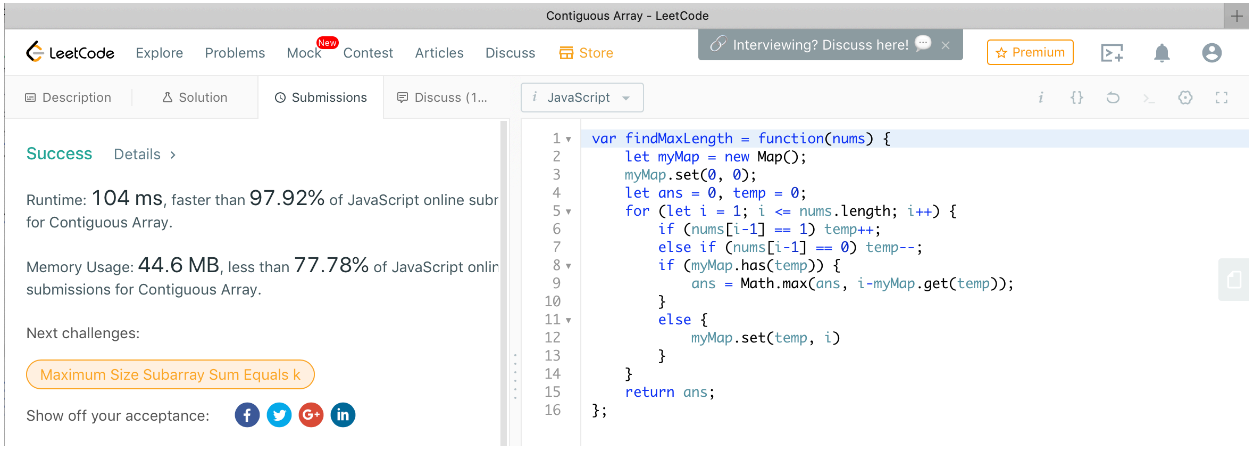
Show off your acceptance:

```

var merge = function(intervals) {
    if (!intervals.length) return [];
    let ans=[];
    intervals.sort(function(a,b) {
        if (a[0] != b[0]) return a[0] - b[0];
        else return a[1] - b[1];
    })
    let prev = intervals[0];
    ans = [prev];
    for (let item of intervals){
        if (item[0] <= prev[1]) {
            prev[1] = Math.max(prev[1], item[1])
        }
        else {
            ans.push(item);
            prev = item;
        }
    }
    return ans;
}

```

Contiguous Array



Success Details ›

Runtime: 104 ms, faster than 97.92% of JavaScript online subr for Contiguous Array.

Memory Usage: 44.6 MB, less than 77.78% of JavaScript onlin submissions for Contiguous Array.

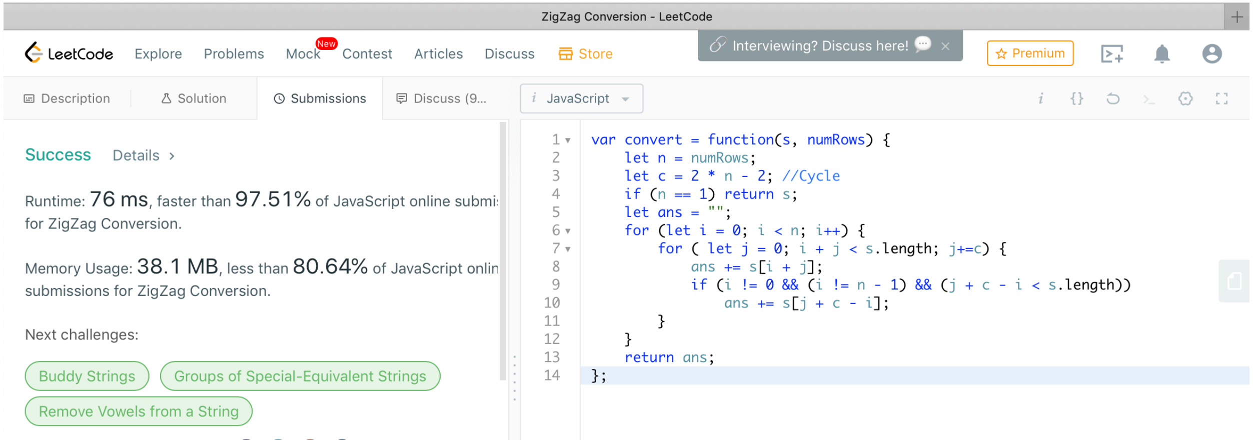
Next challenges:

Maximum Size Subarray Sum Equals k

Show off your acceptance:

```
var findMaxLength = function(nums) {
    let myMap = new Map();
    myMap.set(0, 0);
    let ans = 0, temp = 0;
    for (let i = 1; i <= nums.length; i++) {
        if (nums[i-1] == 1) temp++;
        else if (nums[i-1] == 0) temp--;
        if (myMap.has(temp)) {
            ans = Math.max(ans, i-myMap.get(temp));
        }
        else {
            myMap.set(temp, i)
        }
    }
    return ans;
};
```

ZigZag Conversion



Success Details ›

Runtime: 76 ms, faster than 97.51% of JavaScript online submi for ZigZag Conversion.

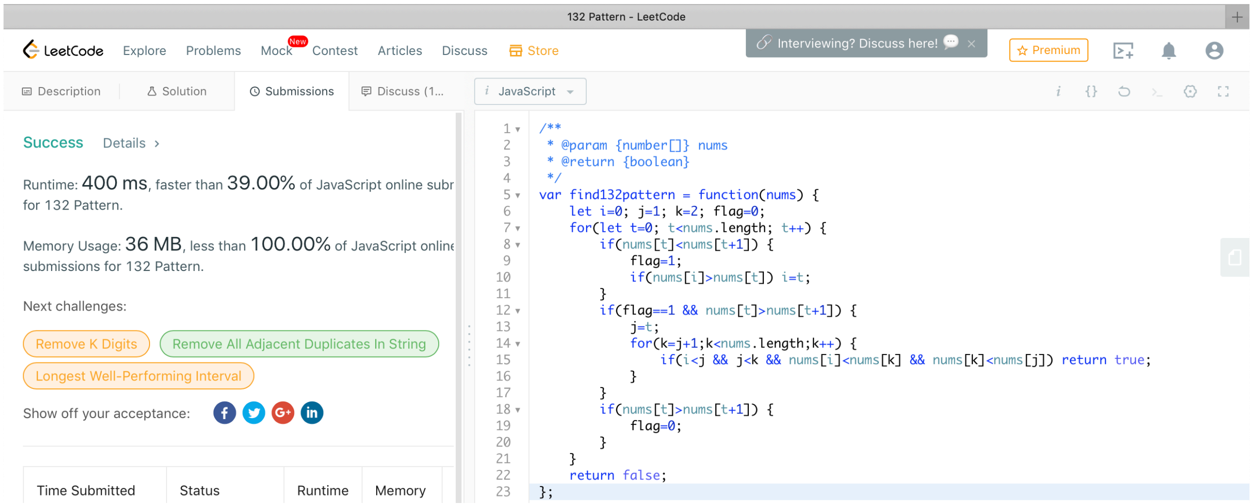
Memory Usage: 38.1 MB, less than 80.64% of JavaScript onlin submissions for ZigZag Conversion.

Next challenges:

Buddy Strings | Groups of Special-Equivalent Strings | Remove Vowels from a String

```
var convert = function(s, numRows) {
    let n = numRows;
    let c = 2 * n - 2; //Cycle
    if (n == 1) return s;
    let ans = "";
    for (let i = 0; i < n; i++) {
        for ( let j = 0; i + j < s.length; j+=c) {
            ans += s[i + j];
            if (i != 0 && (i != n - 1) && (j + c - i < s.length))
                ans += s[j + c - i];
        }
    }
    return ans;
};
```

132-Pattern



Success Details ›

Runtime: 400 ms, faster than 39.00% of JavaScript online subr for 132 Pattern.

Memory Usage: 36 MB, less than 100.00% of JavaScript online submissions for 132 Pattern.

Next challenges:

Remove K Digits
Remove All Adjacent Duplicates In String
Longest Well-Performing Interval
Show off your acceptance:

Time Submitted	Status	Runtime	Memory

```javascript
/**
 * @param {number[]} nums
 * @return {boolean}
 */
var find132pattern = function(nums) {
    let i=0; j=1; k=2; flag=0;
    for(let t=0; t<nums.length; t++) {
        if(nums[t]<nums[t+1]) {
            flag=1;
            if(nums[i]>nums[t]) i=t;
        }
        if(flag==1 && nums[t]>nums[t+1]) {
            j=t;
            for(k=j+1;k<nums.length;k++) {
                if(i<j && j<k && nums[i]<nums[k] && nums[k]<nums[j]) return true;
            }
        }
        if(nums[t]>nums[t+1]) {
            flag=0;
        }
    }
    return false;
};
```