**ARRAYS:**

1. 3Sum:

<https://leetcode.com/problems/3sum/>

class Solution {

    public List<List<Integer>> threeSum(int[] num) {

    Arrays.sort(num);

    List<List<Integer>> res = new LinkedList<>();

    for (int i = 0; i < num.length-2; i++) {

        if (i == 0 || (i > 0 && num[i] != num[i-1])) {

            int lo = i+1, hi = num.length-1, sum = 0 - num[i];

            while (lo < hi) {

                if (num[lo] + num[hi] == sum) {

                    res.add(Arrays.asList(num[i], num[lo], num[hi]));

                    while (lo < hi && num[lo] == num[lo+1]) lo++;

                    while (lo < hi && num[hi] == num[hi-1]) hi--;

                    lo++; hi--;

                } else if (num[lo] + num[hi] < sum) lo++;

                else hi--;

           }

        }

    }

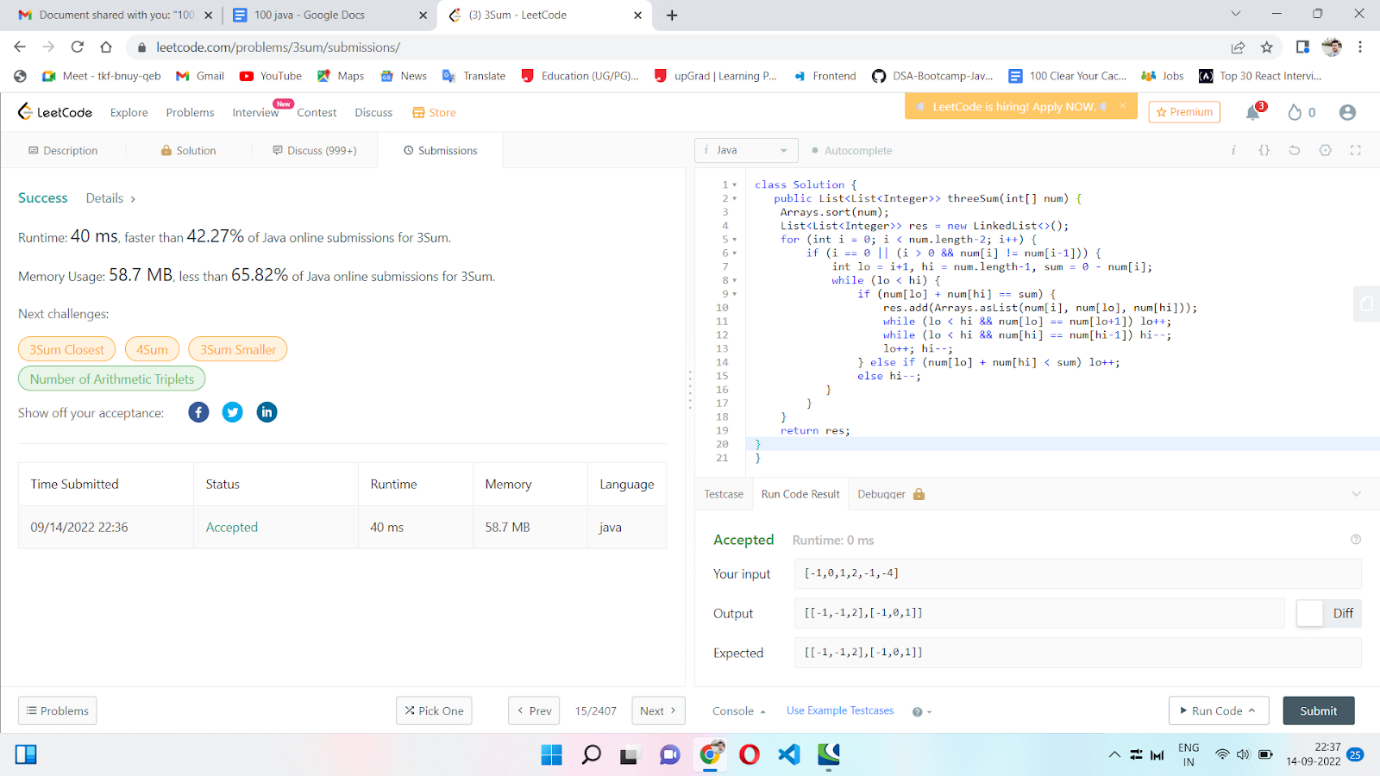
    return res;

}

}

**time complexity is O(N^2)**

**space complexity is O(1)**



2. Move all zeros to the end:

<https://leetcode.com/problems/move-zeroes/>

public class Solution {

   public void moveZeroes(int[] nums)  {

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

        for(int j = 0; j < nums.length-1; j++){

            if(nums[j]==0){

                int temp1 = nums[j];

                nums[j]= nums[j+1];

                nums[j+1] = temp1;

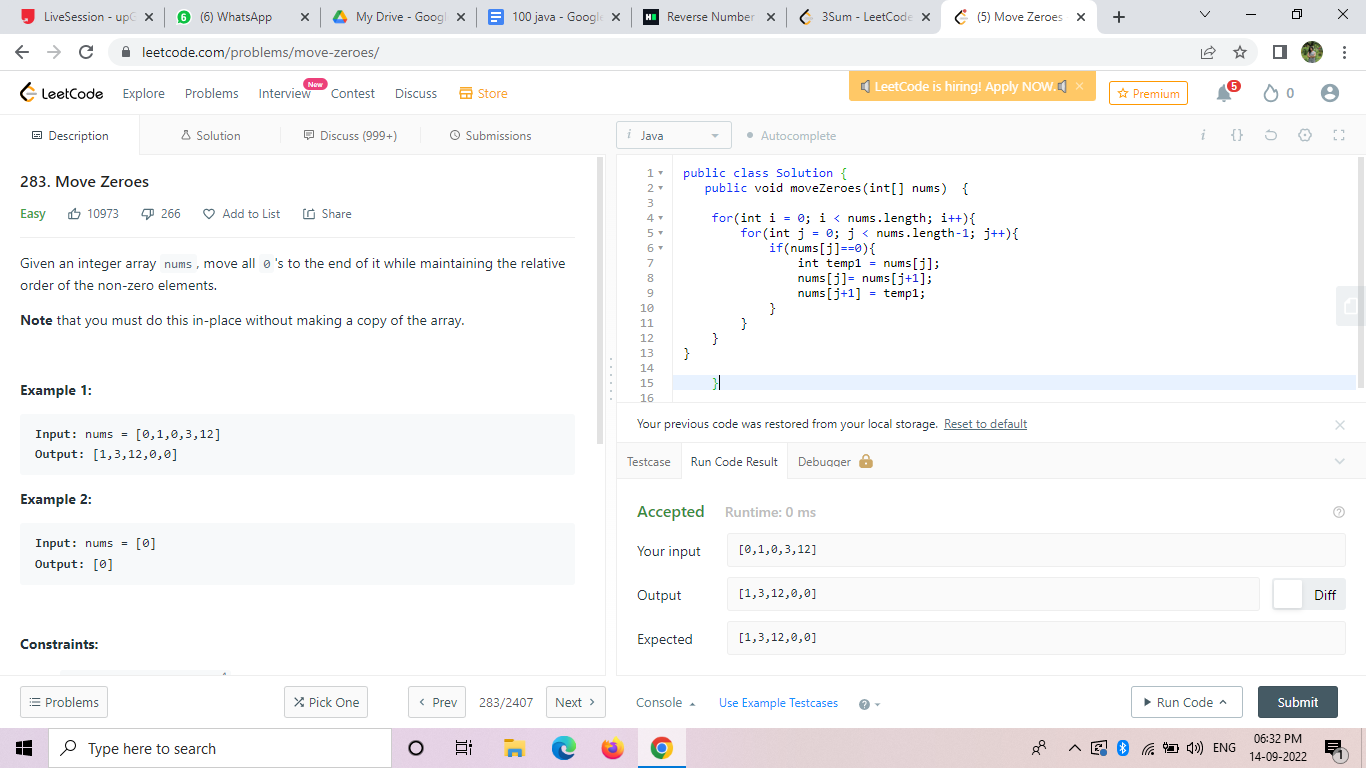
            }

        }

    }

}

    }



3. Search in Rotated Sorted Array:

<https://leetcode.com/problems/search-in-rotated-sorted-array/>

class Solution {

    static int search(int[] nums,int target ){

    int pivot = findPivot(nums);

    if(pivot ==-1){

        return binarySearch(nums,target,0,nums.length-1);

    } if(nums[pivot]==target){

        return pivot;

    }

    if(target>=nums[0]){

        return binarySearch(nums,target,0,pivot-1);

    }

        return binarySearch(nums,target,pivot+1,nums.length-1);

 }

 static int binarySearch(int[] arr, int target, int start,int end){

    while(start<=end){

        int mid=start+(end-start)/2;

        if(target<arr[mid]){

            end=mid-1;

        } else if (target > arr[mid]){

            start=mid+1;

        } else {

            return mid;

        }

    }

    return -1;

 }

 static int findPivot(int[] arr) {

    int start=0;

    int end = arr.length-1;

    while(start<=end){

        int mid = start + (end-start)/2;

        if(mid<end && arr[mid]>arr[mid+1]){

            return mid;

        }

        if (start< mid && arr[mid]<arr[mid-1]){

            return mid-1;

        }

        if(arr[mid]<=arr[start]){

            end=mid-1;

        } else {

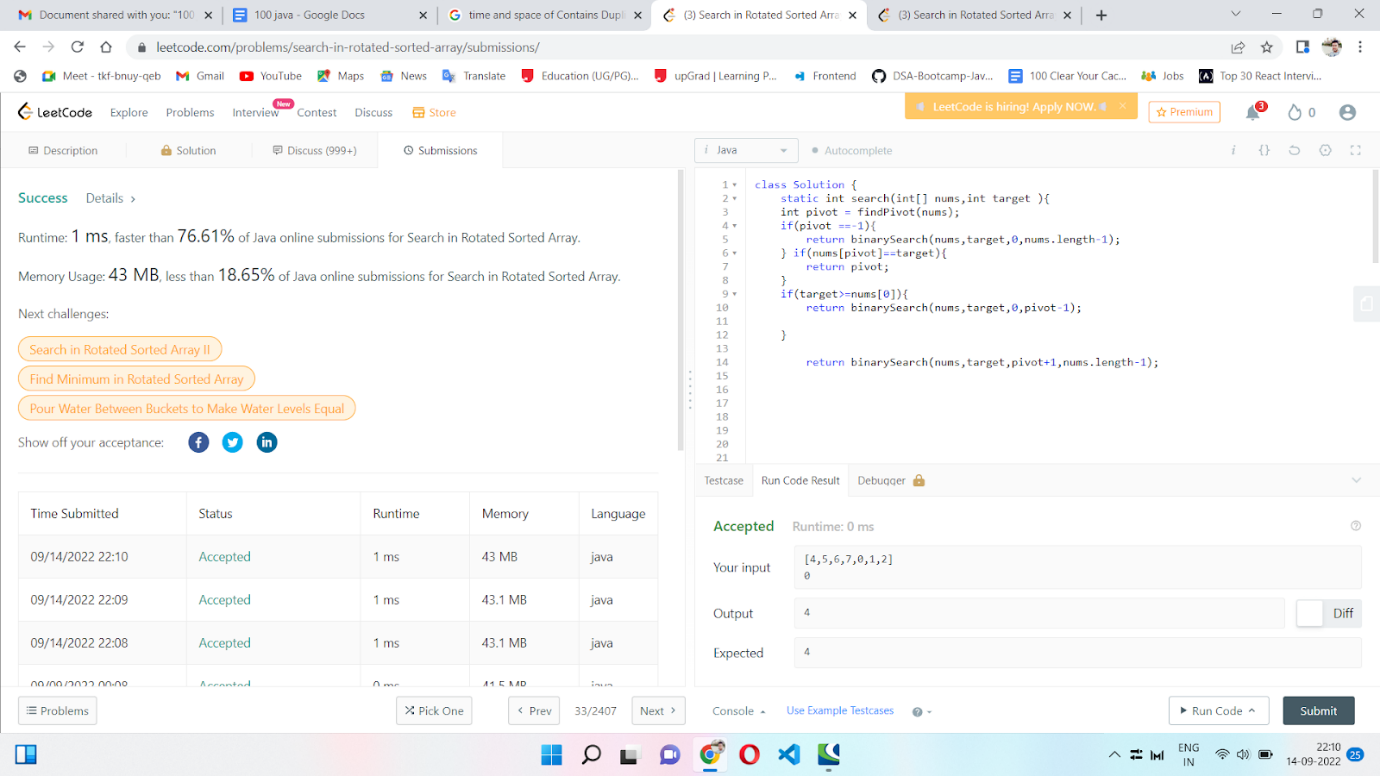
            start= mid+1;

        }

    }  return -1;

    }

}



**Time Complexity:** O(log N)

**Space Complexity:** O(1)

4. Find Peak Element

<https://leetcode.com/problems/find-peak-element/>

class Solution{

    public int findPeakElement(int[] a) {

        int low = 0, mid = 0, high = a.length - 1;

        while(low < high) {

            mid = low + (high-low)/2;

            if(a[mid] < a[mid+1]) low = mid+1;

            else high = mid;

        }

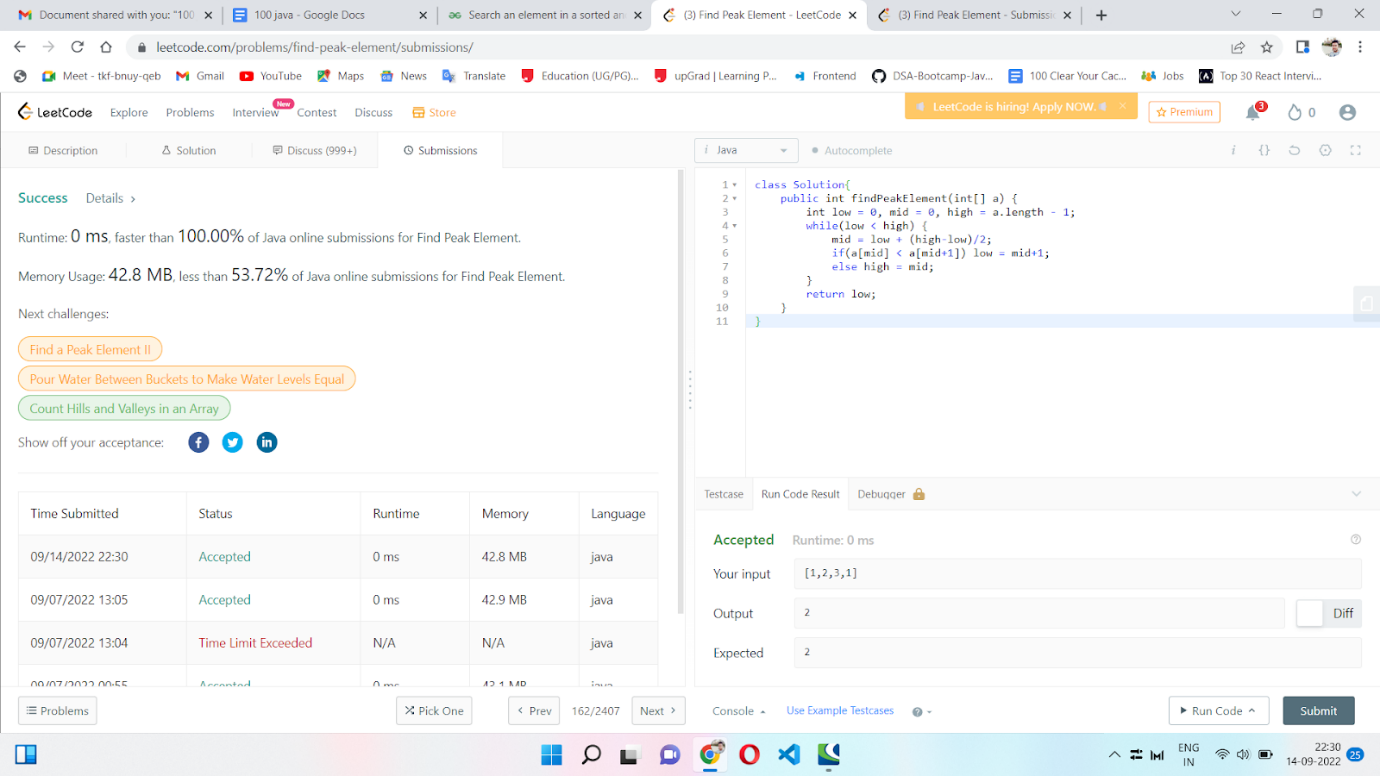
        return low;

    }

}

**Time Complexity:** O(log N)

**Space Complexity:** O(1)



5. Missing Element

<https://leetcode.com/problems/missing-number/>

class Solution {

    public int missingNumber(int[] nums) {

        int n = nums.length;

        int sum= n\*(n+1)/2;

        int sum1=0;

        for(int i=0; i<n;i++){

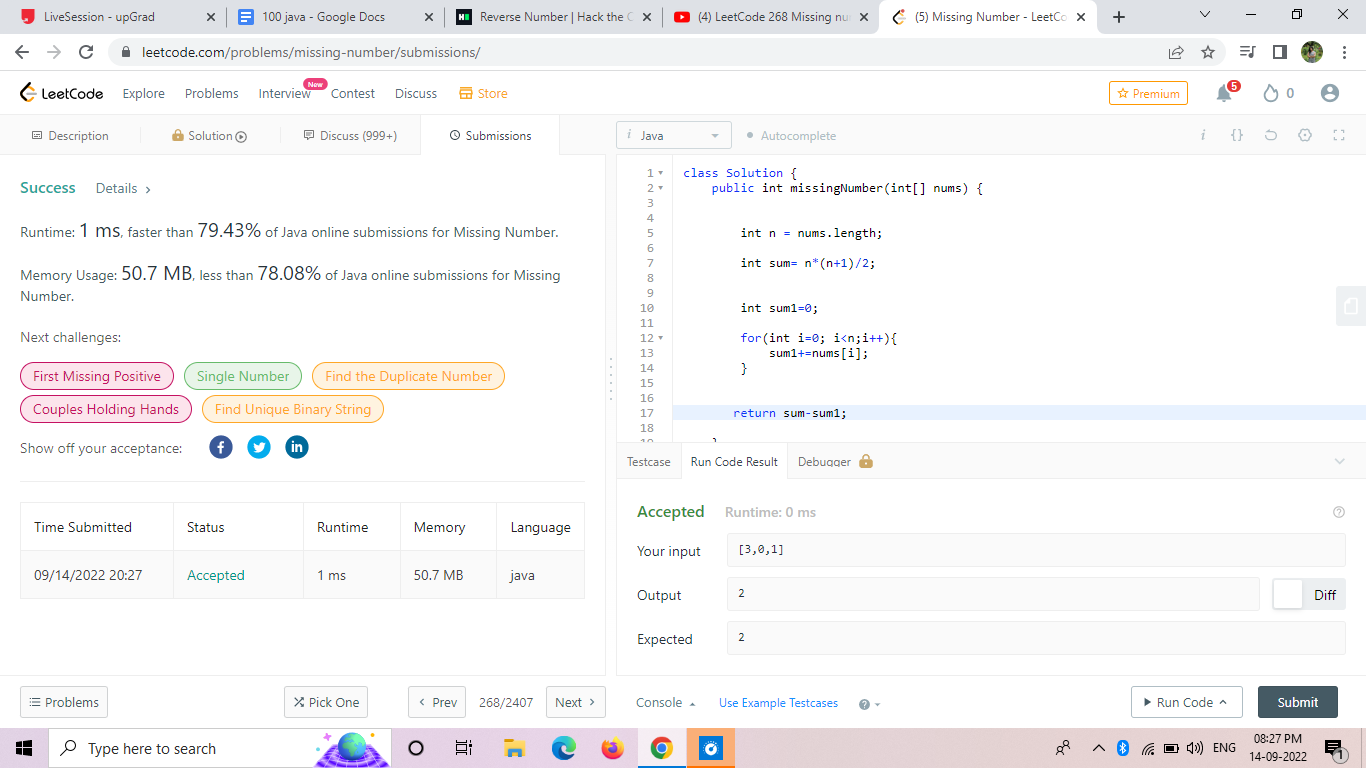
            sum1+=nums[i];

        }

       return sum-sum1;

    }

}



6. Single Number:

<https://leetcode.com/problems/single-number/>

class Solution {

    public int singleNumber(int[] nums) {

    int res=0;

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

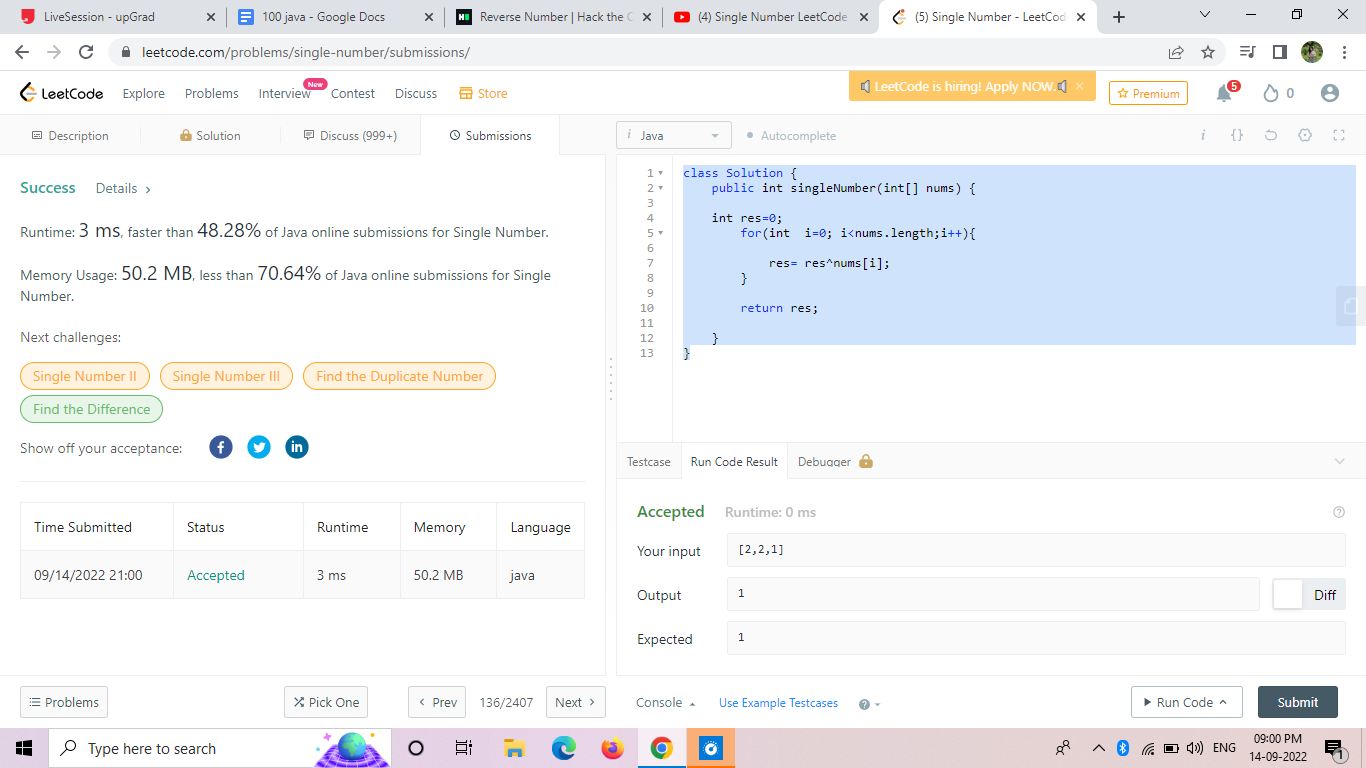
            res= res^nums[i];

        }

        return res;

    }

}



7. Contains Duplicate:

<https://leetcode.com/problems/contains-duplicate/>

class Solution {

    public boolean containsDuplicate(int[] nums) {

        HashSet<Integer> s = new HashSet<>();

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

            if(s.contains(nums[i]))

                return true;

            s.add(nums[i]);

        }

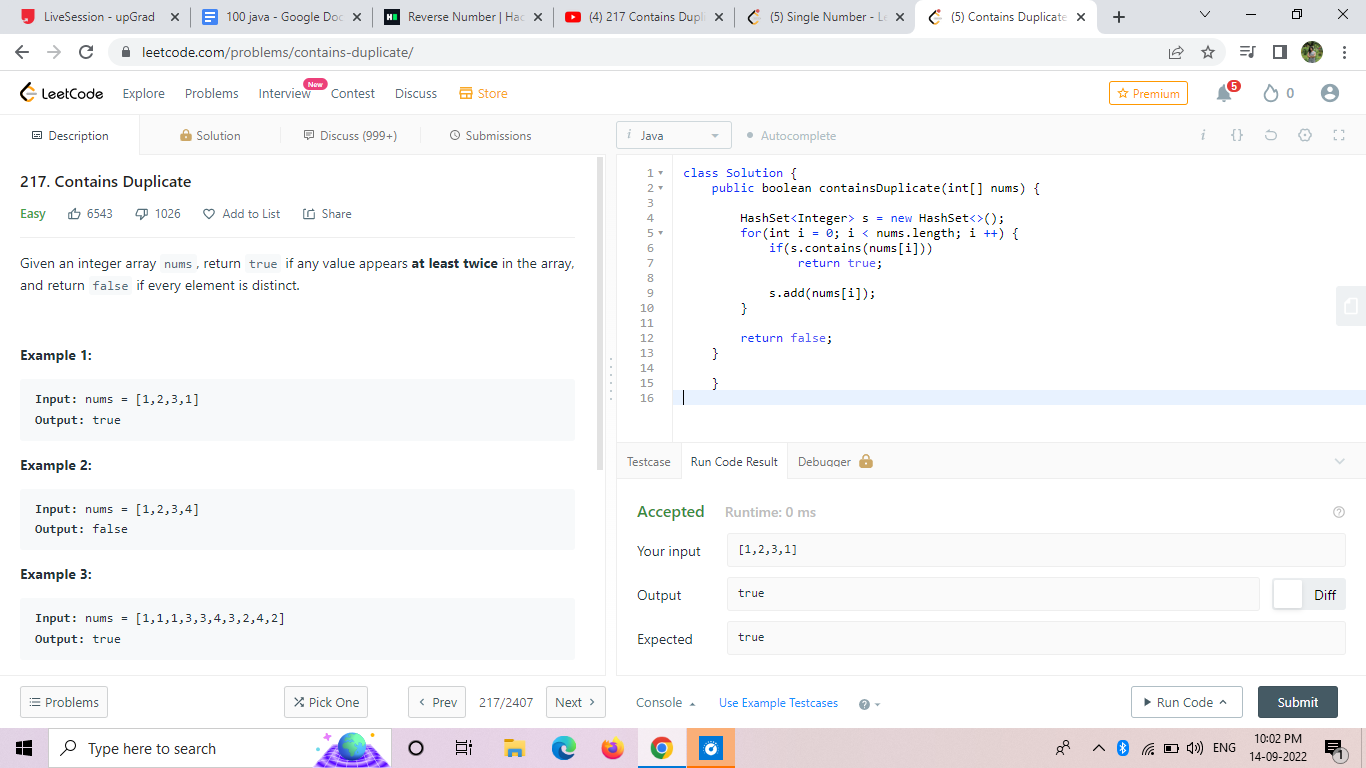
        return false;

    }

    }

Time complexity: O(n)

Space complexity:O(1)



8. Word Search:

[Word Search - LeetCode](https://leetcode.com/problems/word-search/)

public class Solution {

    static boolean[][] visited;

    public boolean exist(char[][] board, String word) {

        visited = new boolean[board.length][board[0].length];

        for(int i = 0; i < board.length; i++){

            for(int j = 0; j < board[i].length; j++){

                if((word.charAt(0) == board[i][j]) && search(board, word, i, j, 0)){

                    return true;

                }

            }

        }

        return false;

    }

    private boolean search(char[][]board, String word, int i, int j, int index){

        if(index == word.length()){

            return true;

        }

        if(i >= board.length || i < 0 || j >= board[i].length || j < 0 || board[i][j] != word.charAt(index) || visited[i][j]){

            return false;

        }

        visited[i][j] = true;

        if(search(board, word, i-1, j, index+1) ||

           search(board, word, i+1, j, index+1) ||

           search(board, word, i, j-1, index+1) ||

           search(board, word, i, j+1, index+1)){

            return true;

        }

        visited[i][j] = false;

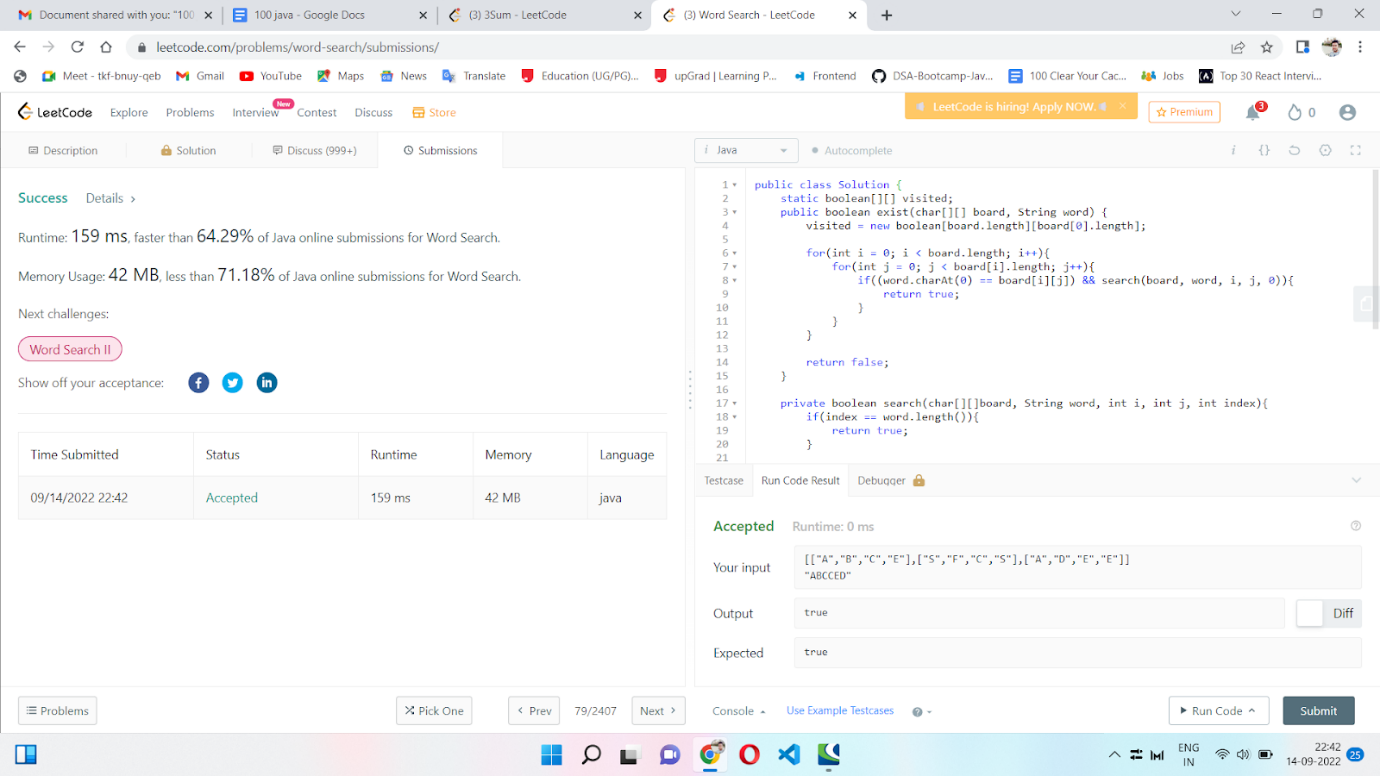
        return false;

    }

}

Time complexity: O(mnv)

Space complexity:O(1)



9. Find all disappeared Numbers in an array:

<https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array/>

class Solution {

  public List<Integer> findDisappearedNumbers(int[] nums) {

        List<Integer> list = new ArrayList<>();

        int idx = -1;

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

            if(nums[i] < 0){

                idx = nums[i]\*-1-1;

            }else{

                idx = nums[i]-1;

            }

            if(nums[idx]>0){

                nums[idx] = -nums[idx];

            }

        }

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

            if(nums[i] > 0){

                list.add(i+1);

            };

        }

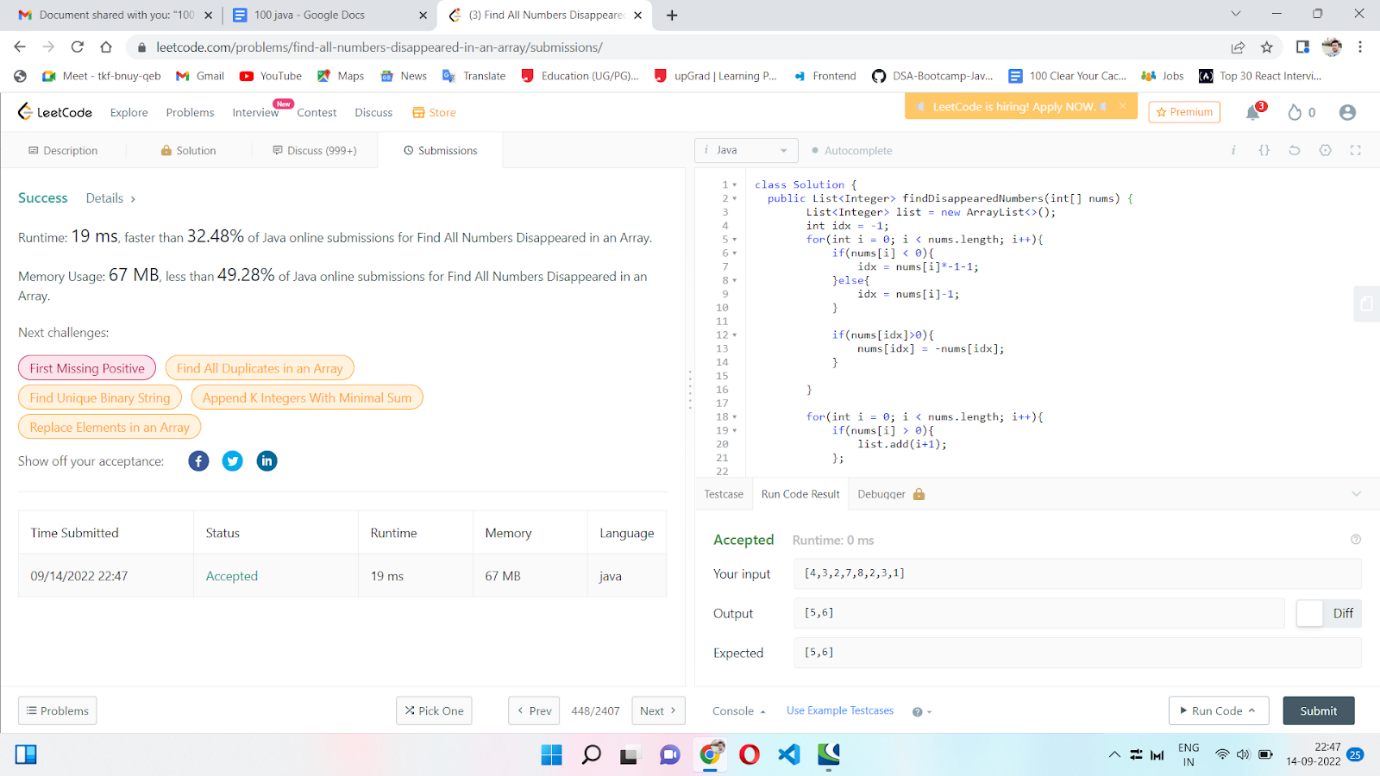
        return list;

    }

}

Time complexity :O(n)

Space complexity:O(1)



10. Product of Array Except Self

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

class Solution {

  public int[] productExceptSelf(int[] nums) {

    int[] result = new int[nums.length];

    for (int i = 0, tmp = 1; i < nums.length; i++) {

        result[i] = tmp;

        tmp \*= nums[i];

    }

    for (int i = nums.length - 1, tmp = 1; i >= 0; i--) {

        result[i] \*= tmp;

        tmp \*= nums[i];

    }

    return result;

}

}

Time complexity :O(n)

Space complexity:O(1)

