**Assignment -6**

**Name: Ravi Prakash Singh Uid: 22BCS14299**

**Section- FL\_IOT\_602/A**

[**108. Convert Sorted Array to Binary Search Tree**](https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/)

// Definition for a binary tree node.

public class TreeNode {

    int val;

    TreeNode left;

    TreeNode right;

    TreeNode() {}

    TreeNode(int val) { this.val = val; }

    TreeNode(int val, TreeNode left, TreeNode right) {

        this.val = val;

        this.left = left;

        this.right = right;

    }

}

class Solution {

    public TreeNode sortedArrayToBST(int[] nums) {

        return helper(nums, 0, nums.length - 1);

    }

    private TreeNode helper(int[] nums, int left, int right) {

        if (left > right) return null;

        int mid = (left + right) / 2;

        TreeNode root = new TreeNode(nums[mid]);

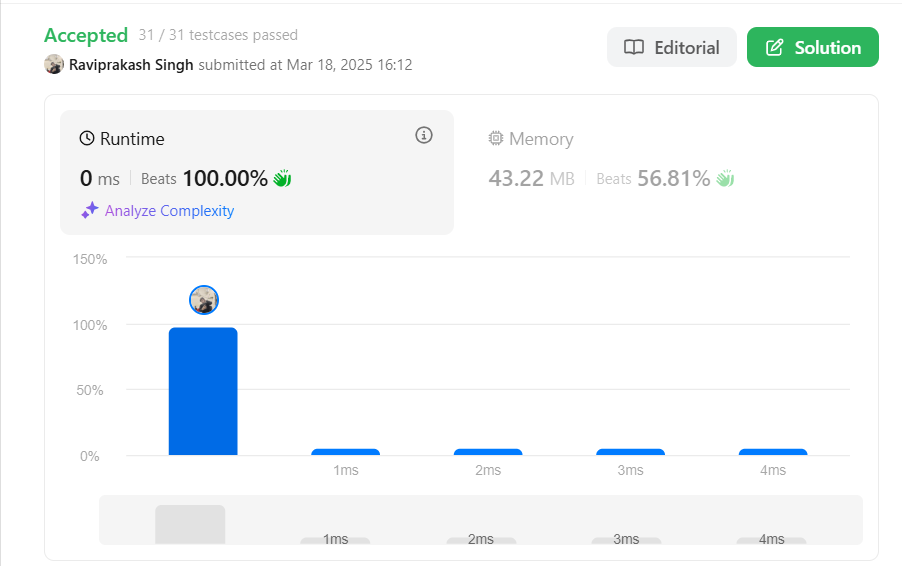
        root.left = helper(nums, left, mid - 1);

        root.right = helper(nums, mid + 1, right);

        return root;

    }

}



[**191. Number of 1 Bits**](https://leetcode.com/problems/number-of-1-bits/)

class Solution {

    public int hammingWeight(int n) {

        int count = 0;

        while (n != 0) {

            count += (n & 1);

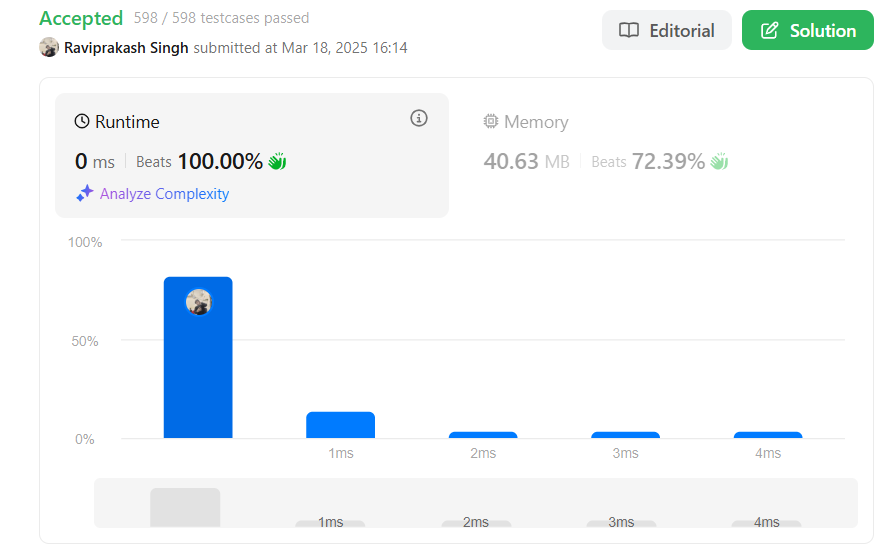
            n >>>=1;

        }

        return count;

    }

}



[**912. Sort an Array**](https://leetcode.com/problems/sort-an-array/)

class Solution {

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

mergeSort(nums, 0, nums.length - 1);

return nums;

}

private void mergeSort(int[] array, int low, int high) {

if (low >= high) {

return;

}

int mid = low + (high - low) / 2;

mergeSort(array, low, mid);

mergeSort(array, mid + 1, high);

merge(array, low, mid, high);

}

private void merge(int[] array, int low, int mid, int high) {

int n1 = mid - low + 1;

int n2 = high - mid;

int[] leftPart = new int[n1];

int[] rightPart = new int[n2];

System.arraycopy(array, low, leftPart, 0, n1);

System.arraycopy(array, mid + 1, rightPart, 0, n2);

int p1 = 0, p2 = 0, writeInd = low;

while (p1 < n1 && p2 < n2) {

if (leftPart[p1] <= rightPart[p2]) {

array[writeInd++] = leftPart[p1++];

} else {

array[writeInd++] = rightPart[p2++];

}

}

while (p1 < n1) {

array[writeInd++] = leftPart[p1++];

}

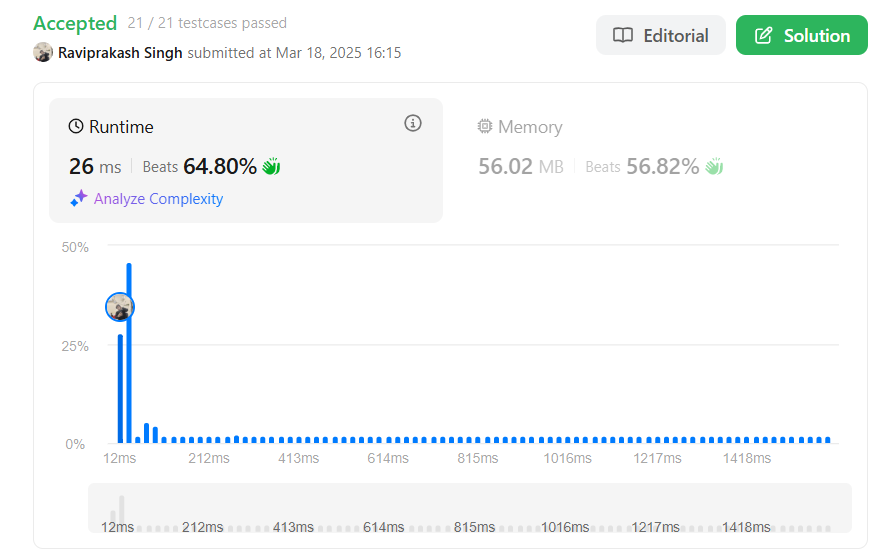
while (p2 < n2) {

array[writeInd++] = rightPart[p2++];

}

}

}



[**53. Maximum Subarray**](https://leetcode.com/problems/maximum-subarray/)

class Solution {

    public int maxSubArray(int[] nums) {

        int max = nums[0];

        int currSum = 0;

        for(int i : nums){

                currSum+=i;

                max = max<currSum?currSum:max;

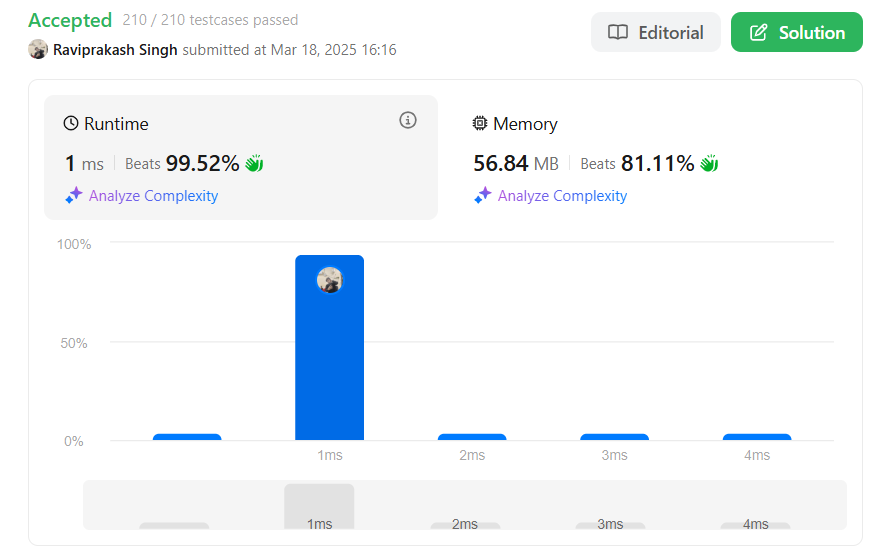
                if(currSum<0)currSum=0;

        }

        return max;

    }

}



[**932. Beautiful Array**](https://leetcode.com/problems/beautiful-array/)

class Solution {

    public int[] beautifulArray(int n) {

        return buildBeautifulArray(n);

    }

    private int[] buildBeautifulArray(int n) {

        if (n == 1) {

            return new int[] { 1 };  // Base case

        }

        int[] odd = buildBeautifulArray((n + 1) / 2);  // For odd numbers

        int[] even = buildBeautifulArray(n / 2);  // For even numbers

        int[] result = new int[n];

        int idx = 0;

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

            result[idx++] = 2 \* odd[i] - 1;

        }

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

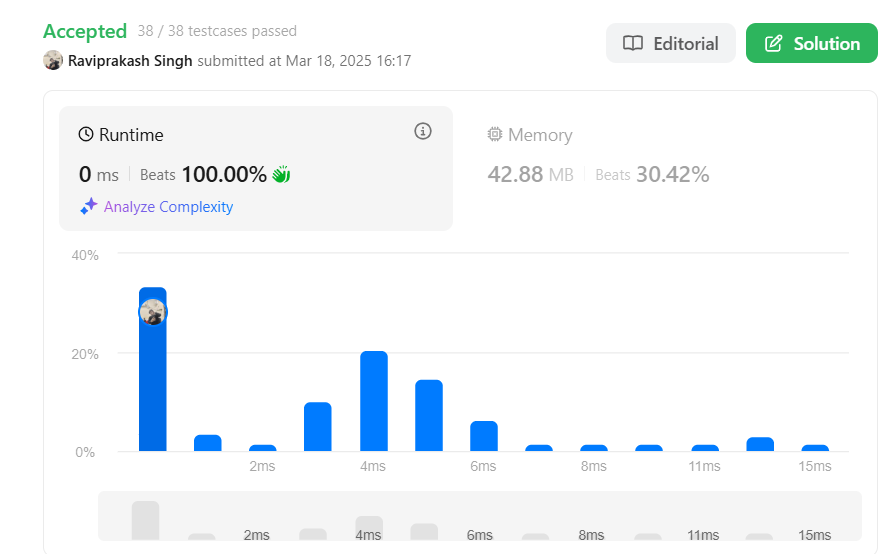
            result[idx++] = 2 \* even[i];

        }

        return result;

    }

}



[**372. Super Pow**](https://leetcode.com/problems/super-pow/)

class Solution {

    private static final int MOD = 1337;

    private int modPow(int a, int b) {

        int result = 1;

        a = a % MOD;

        while (b > 0) {

            if (b % 2 == 1) {

                result = (result \* a) % MOD;

            }

            a = (a \* a) % MOD;

            b /= 2;

        }

        return result;

    }

    public int superPow(int a, int[] b) {

        int result = 1;

        a = a % MOD;

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

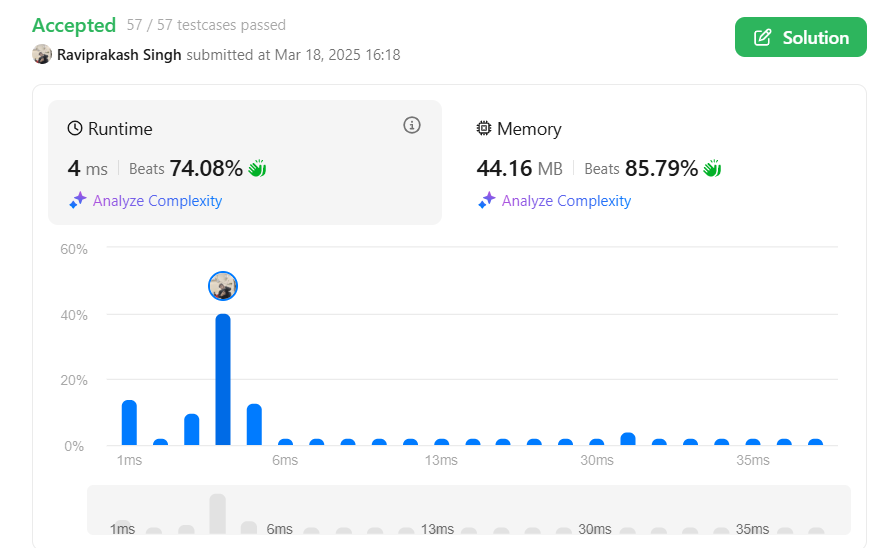
            result = modPow(result, 10) \* modPow(a, b[i]) % MOD;

        }

        return result;

    }

}



[**218. The Skyline Problem**](https://leetcode.com/problems/the-skyline-problem/)

class Solution {

public List<List<Integer>> getSkyline(int[][] buildings) {

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

List<int[]> lines = new ArrayList<>();

for (int[] building: buildings) {

lines.add(new int[] {building[0], building[2]});

lines.add(new int[] {building[1], -building[2]});

}

Collections.sort(lines, (a, b)->a[0]==b[0]?b[1]-a[1]:a[0]-b[0]);

TreeMap<Integer, Integer> map = new TreeMap<>();

map.put(0, 1);

int prev=0;

for (int[] line: lines) {

if (line[1]>0) {

map.put(line[1], map.getOrDefault(line[1], 0)+1);

} else {

int f = map.get(-line[1]);

if (f==1) map.remove(-line[1]);

else map.put(-line[1], f-1);

}

int curr = map.lastKey();

if (curr!=prev) {

list.add(Arrays.asList(line[0], curr));

prev=curr;

}

}

return list;

}

}