**Assignment 6**

**Name: Aditi Singh UID: 22BCS10670**

**Section: FL\_IOT\_602 Group: A**

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

**Sol:** 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);

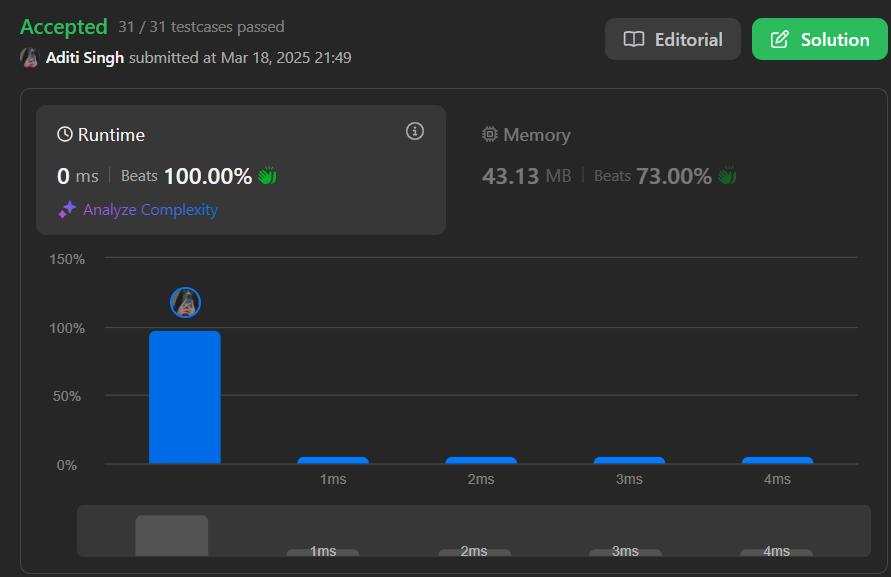
        root.right = helper(nums, mid + 1, right);

        return root;

    }

}

**Output:**

****

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

**Sol:** public class Solution {

    public int hammingWeight(int n) {

        int res = 0;

        for (int i = 0; i < 32; i++) {

            if (((n >> i) & 1) == 1) {

                res += 1;

            }

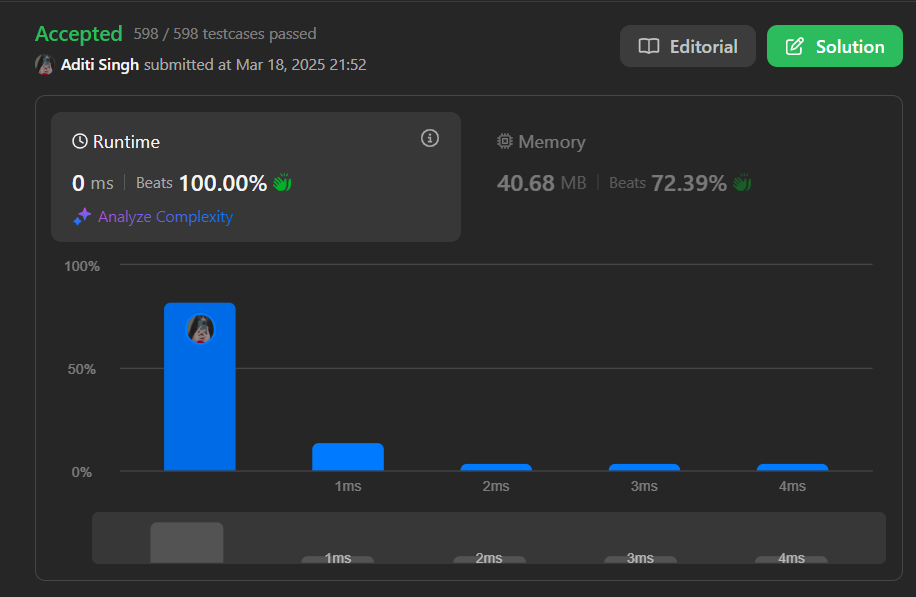
        }

        return res;

    }

}

**Output:**

****

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

**Sol:** class Solution {

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

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

return arr;

}

public static void mergeSort(int start,int end,int []arr){

if (start>=end){

return;

}

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

mergeSort(start,mid,arr);

mergeSort(mid+1,end,arr);

merge(arr,start,mid,end);

}

public static void merge(int[]arr,int start,int mid,int end){

int []temp = new int[end-start+1];

int i = start;

int j = mid+1;

int k = 0;

while(i<=mid && j<=end){

if(arr[i]<arr[j]){

temp[k]=arr[i];

i++;

}

else{

temp[k]=arr[j];

j++;

}

k++;

}

while(i<=mid){

temp[k++]=arr[i++];

}

while(j<=end){

temp[k++]=arr[j++];

}

for(k=0,i=start;k< temp.length;k++,i++){

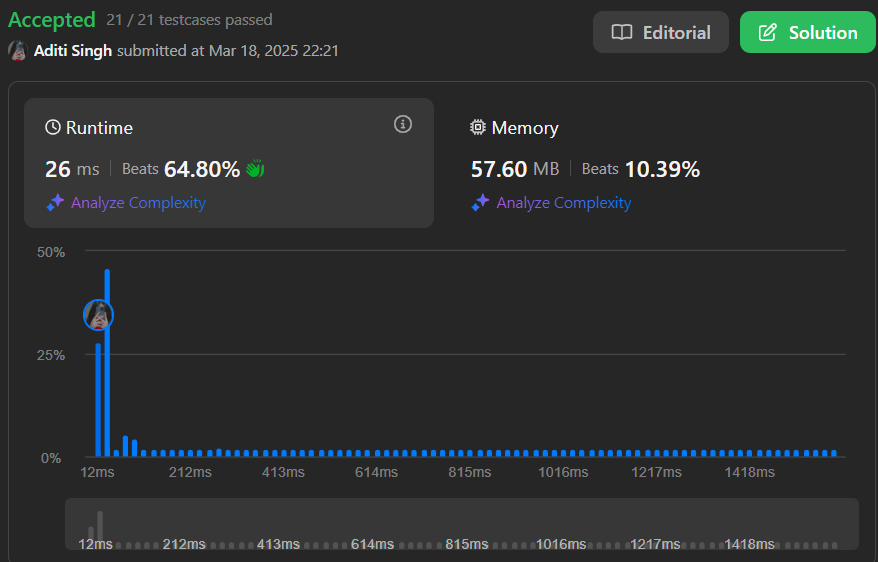
arr[i]=temp[k];

}

}

}

**Output:**

****

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

**Sol:**

class Solution {

    public int maxSubArray(int[] nums) {

        int max=Integer.MIN\_VALUE;

        int c=0;

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

         int num = nums[i];

        if (c < 0) {

        c = 0;

    }

      c=c+ num;

     max = Math.max(max, c);

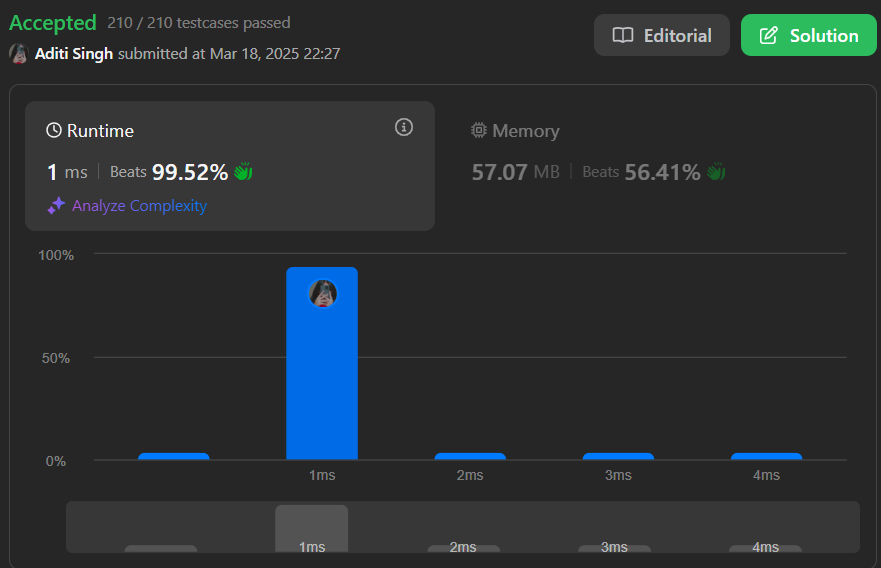
}

        return max;

    }

}

**Output:**

****

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

**Sol:** class Solution {

    public int[] beautifulArray(int N) {

        int[] res = new int[N];

        if (N == 1)

        {

            return new int[] {1};

        }

        else if (N == 2) {

            return new int[] {1, 2};

        }

        else{

            int[] odds = beautifulArray((N + 1) / 2);

            int[] even = beautifulArray(N / 2);

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

            {

                res[i] = odds[i] \* 2 - 1;

            }

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

            {

                res[odds.length + j] = even[j] \* 2;

            }

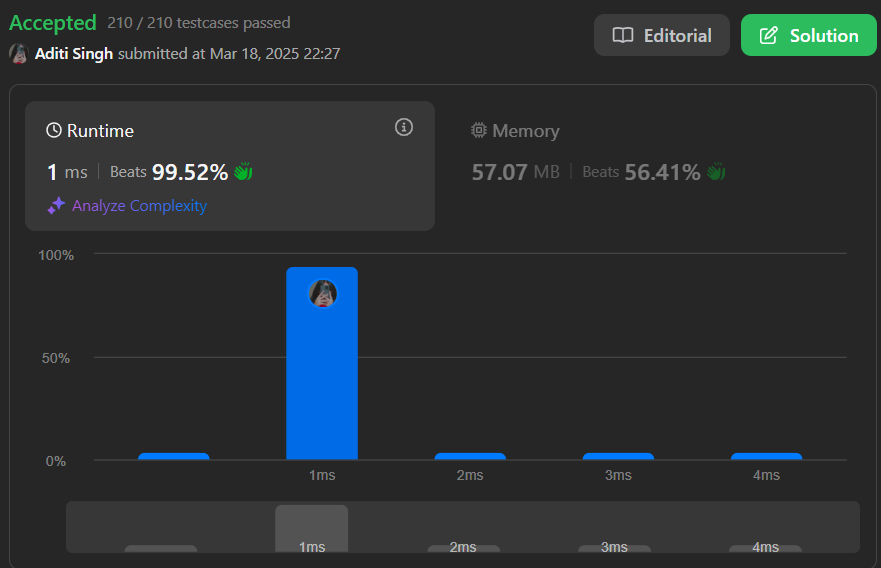
        }

        return res;

    }

}

**Output:**

****

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

**Sol:** class Solution {

    private int binExp(int a, int b, int M) {

        int res = 1;

        a %= M;

        while (b > 0) {

            if ((b & 1) != 0)

                res = (res \* a) % M;

            a = (a \* a) % M;

            b >>= 1;

        }

        return res;

    }

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

        int m = 1140;

        int exp = 0;

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

            exp = (exp \* 10 + b[i]) % m;

        if (exp == 0)

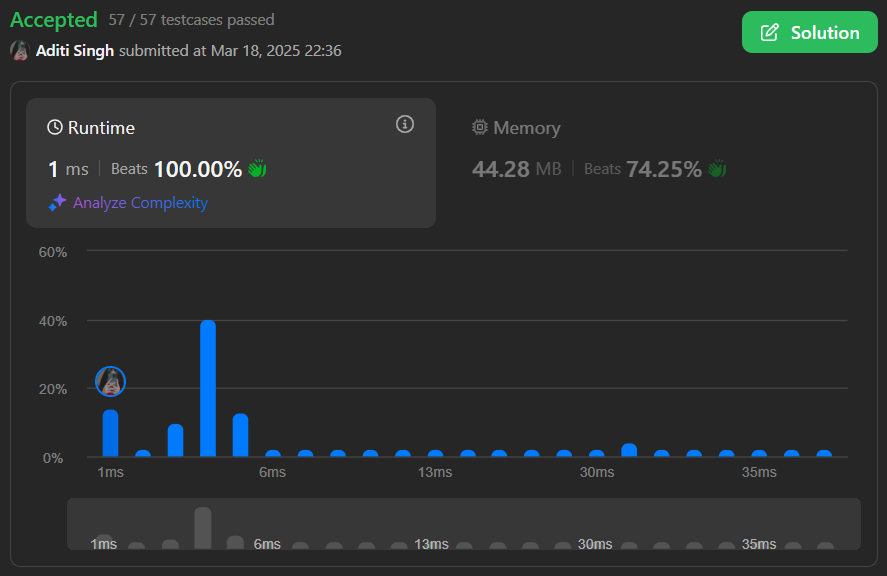
            exp = 1140;

        return binExp(a, exp, 1337);

    }

}

**Output:**

****

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

**Sol:**

class Solution {

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

return new AbstractList<List<Integer>>() {

private List<List<Integer>> resList;

private void onload() {

resList = new ArrayList<>();

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

for (int[] building : buildings) {

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

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

}

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

PriorityQueue<Integer> pq = new PriorityQueue<>((a, b) -> b - a);

pq.offer(0);

int prev = 0;

for (int[] height : heights) {

if (height[1] < 0) {

pq.offer(-height[1]);

} else {

pq.remove(height[1]);

}

int cur = pq.peek();

if (prev != cur) {

resList.add(Arrays.asList(height[0], cur));

prev = cur;

}

}

}

private void init() {

if (null == resList) {

onload();

}

}

@Override

public List<Integer> get(int index) {

init();

return resList.get(index);

}

@Override

public int size() {

init();

return resList.size();

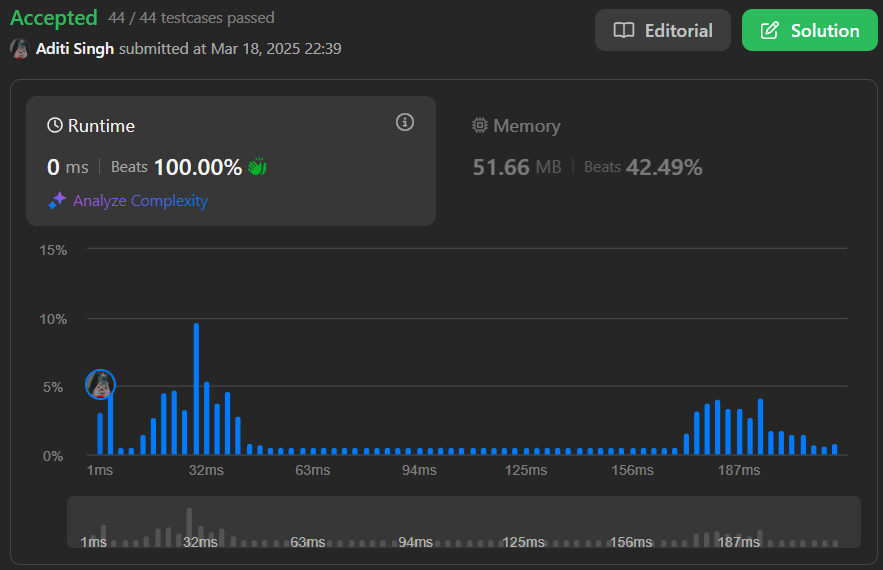
}

};

}

}

**Output:**

****