**Assignment 6**

**Name – Aditya Taragi Uid – 22BCS10422.**

**Problem 1: Convert Sorted Array to Binary Search Tree (**<https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/> **)**

**Code:**class Solution {

public:

    TreeNode\* sortedArrayToBST(vector<int>& nums) {

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

    }

private:

    TreeNode\* helper(vector<int>& nums, int left, int right) {

        if (left > right) return nullptr;

        int mid = left + (right - left) / 2;

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

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

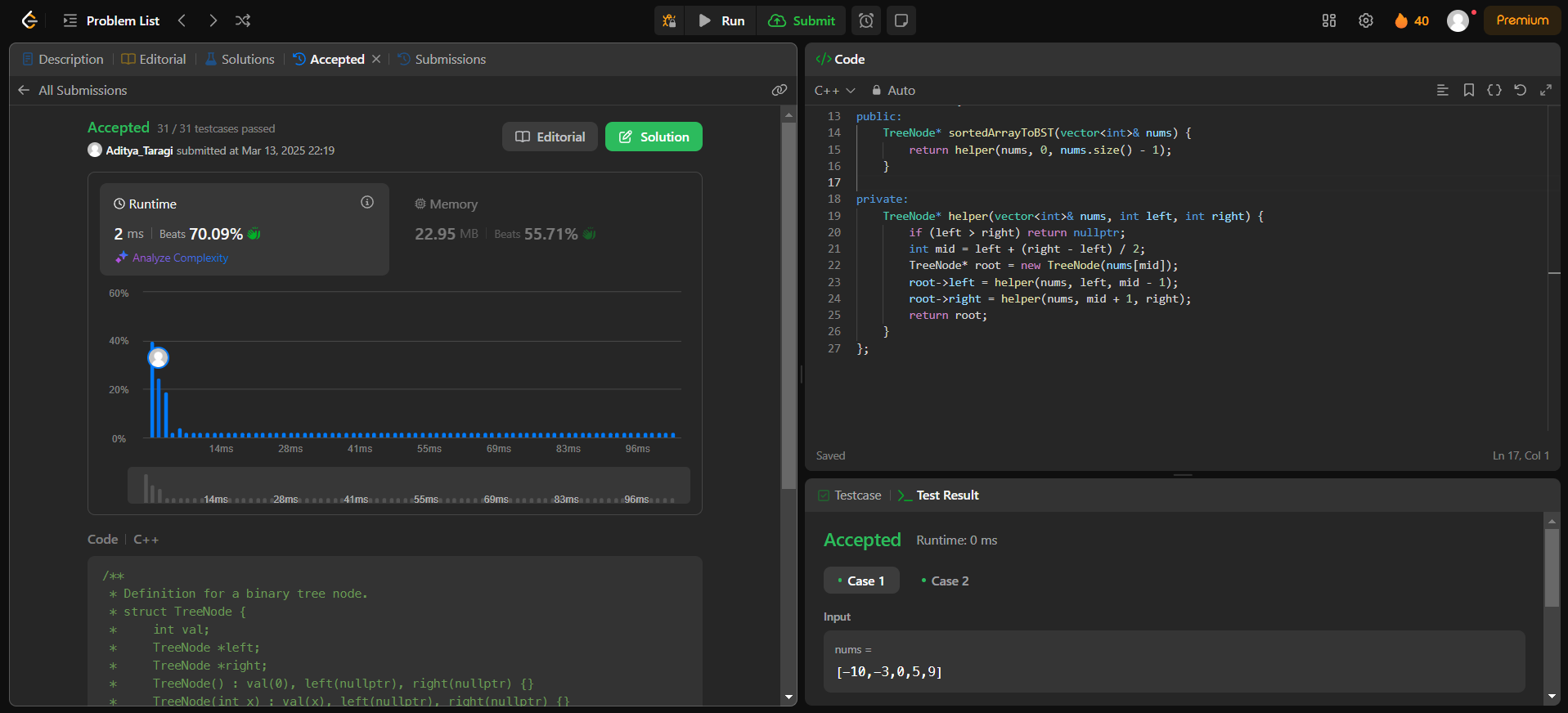
        root->right = helper(nums, mid + 1, right);

        return root;

    }

};

**Screenshot:**

****

**Problem 2: Number of 1 Bits (**<https://leetcode.com/problems/number-of-1-bits/> **)**

**Code:**class Solution {

public:

    int hammingWeight(int n) {

        int count = 0;

        while(n) {

            n = n & (n-1);

            count++;

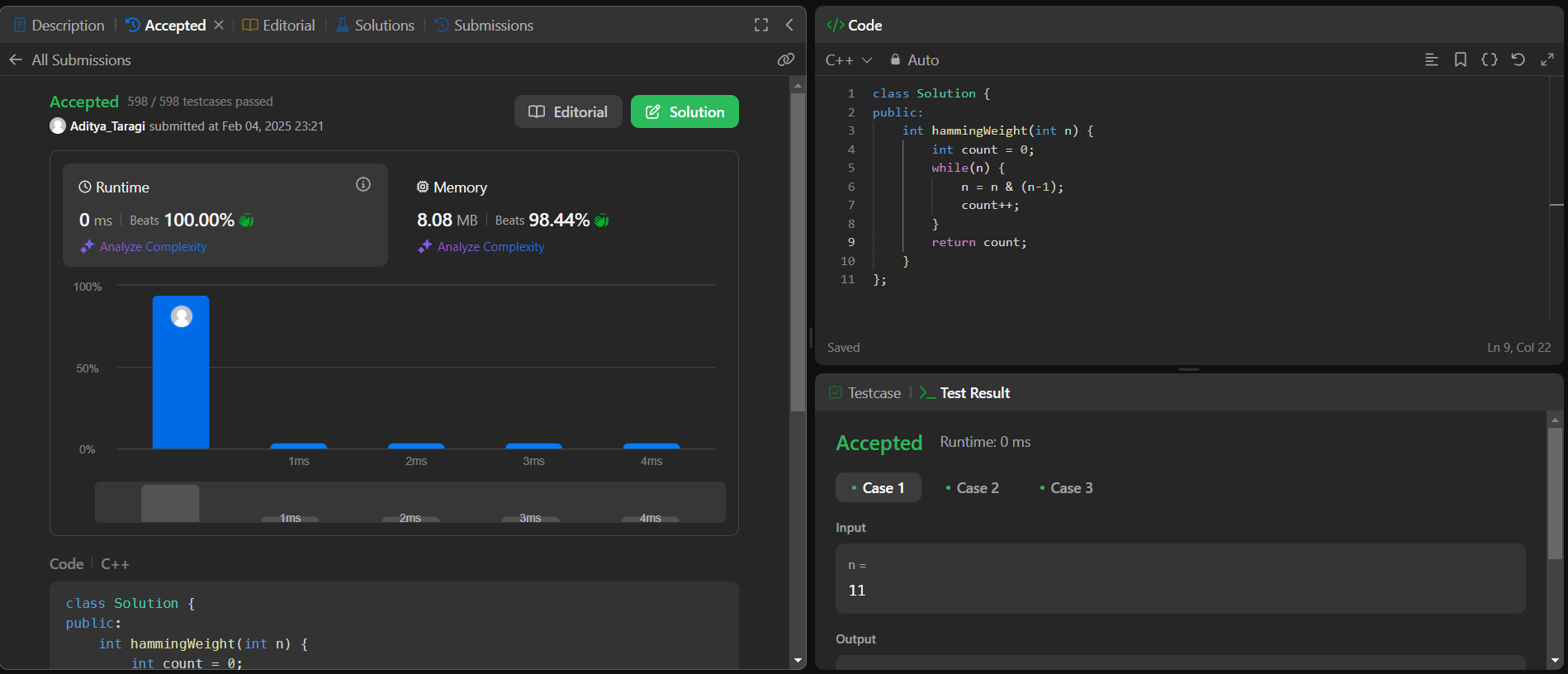
        }

        return count;

    }

};

**Screenshot:**



**Problem 3: Sort an Array (**<https://leetcode.com/problems/sort-an-array/> **)**

**Code:**class Solution {

public:

    void quickSort(vector<int>& nums, int left, int right) {

        if (left >= right) return;

        int pivotIndex = left + rand() % (right - left + 1);

        swap(nums[pivotIndex], nums[right]);

        int pivot = nums[right];

        int partitionIndex = left;

        for (int i = left; i < right; i++) {

            if (nums[i] < pivot) {

                swap(nums[i], nums[partitionIndex]);

                partitionIndex++;

            }

        }

        swap(nums[partitionIndex], nums[right]);

        quickSort(nums, left, partitionIndex - 1);

        quickSort(nums, partitionIndex + 1, right);

    }

    vector<int> sortArray(vector<int>& nums) {

        srand(time(0));

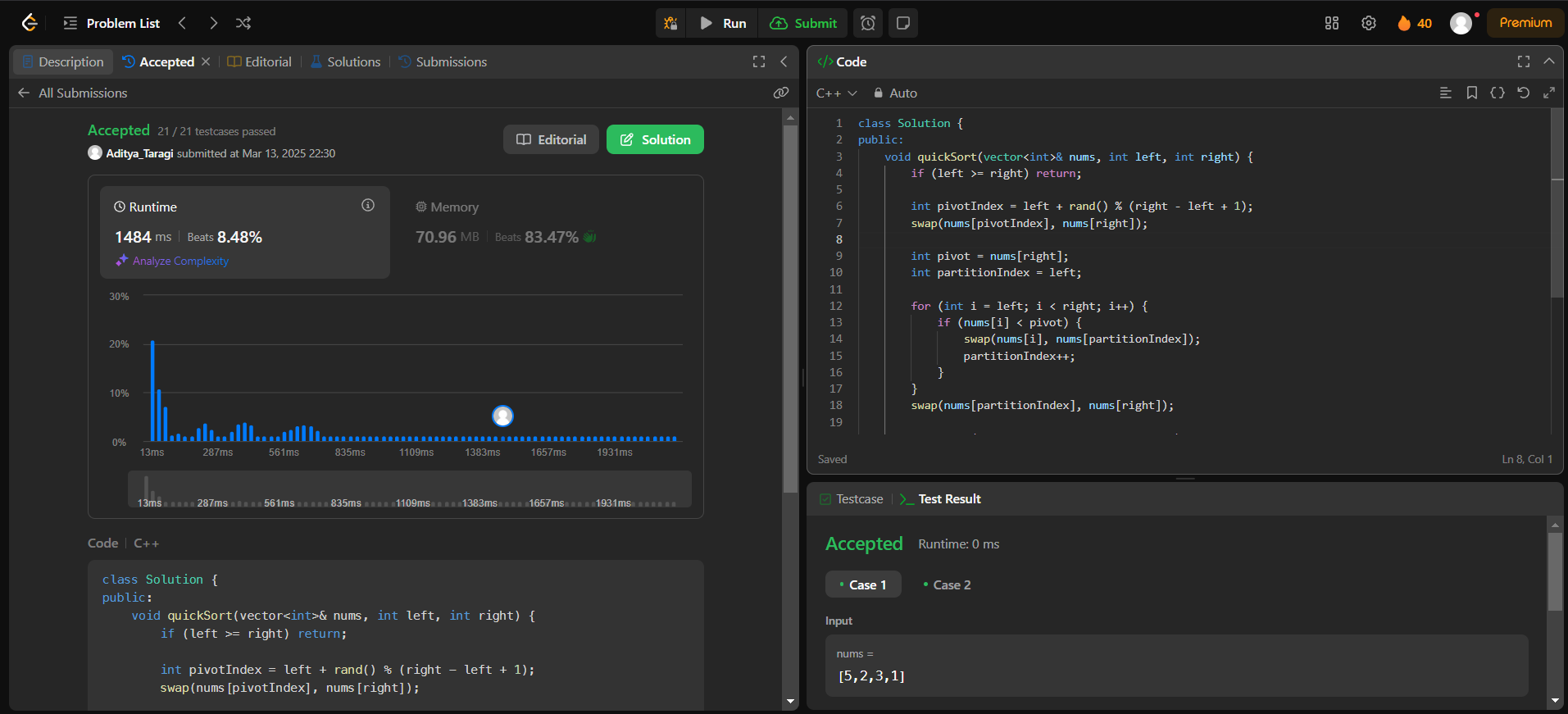
        quickSort(nums, 0, nums.size() - 1);

        return nums;

    }

};

**Screenshot:**

****

**Problem 4: Maximum Subarray (**<https://leetcode.com/problems/maximum-subarray/> **)**

**Code:**class Solution {

public:

    void quickSort(vector<int>& nums, int left, int right) {

        if (left >= right) return;

        int pivot = nums[right];

        int partitionIndex = left;

        for (int i = left; i < right; i++) {

            if (nums[i] < pivot) {

                swap(nums[i], nums[partitionIndex]);

                partitionIndex++;

            }

        }

        swap(nums[partitionIndex], nums[right]);

        quickSort(nums, left, partitionIndex - 1);

        quickSort(nums, partitionIndex + 1, right);

    }

    vector<int> sortArray(vector<int>& nums) {

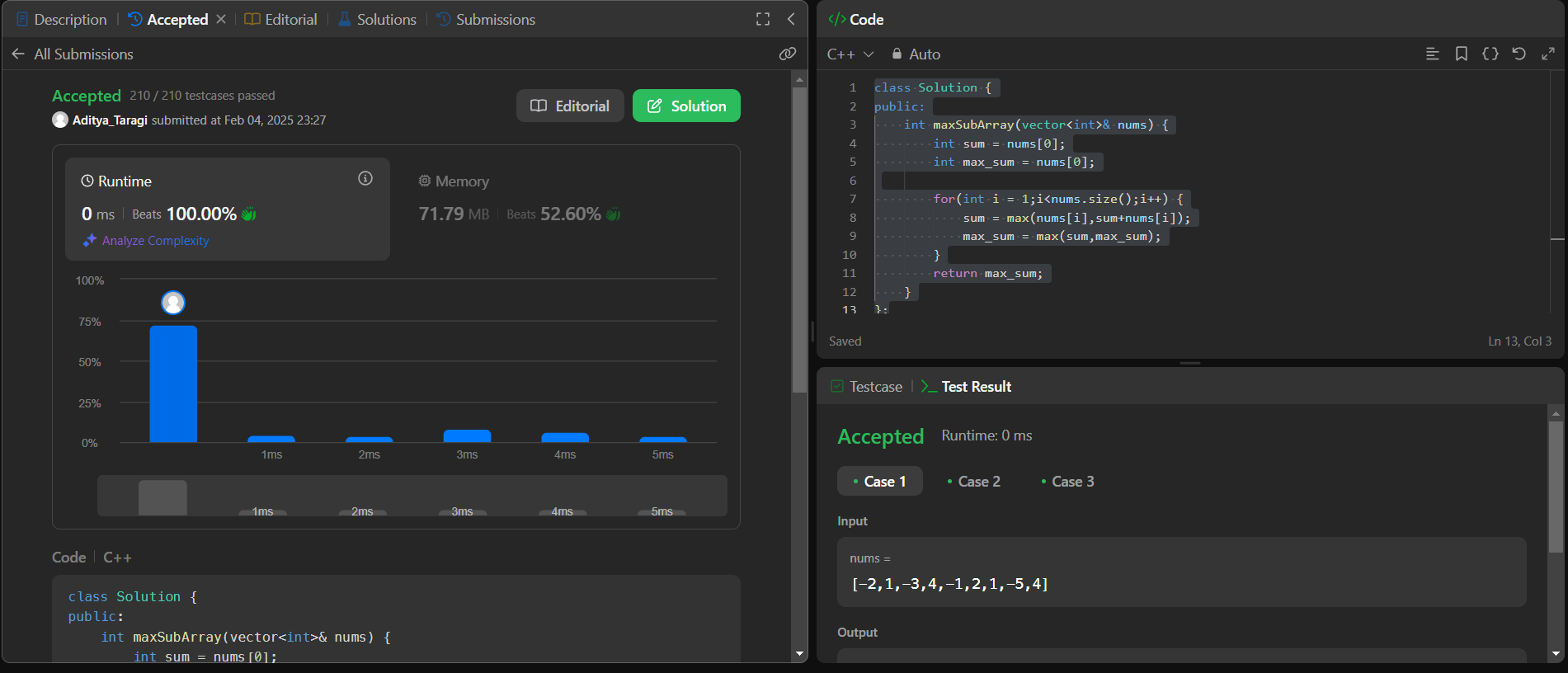
        quickSort(nums, 0, nums.size() - 1);

        return nums;

    }

};

**Screenshot:**



**Problem 5: Beautiful Array (**<https://leetcode.com/problems/beautiful-array/> **)**

**Code:**class Solution {

public:

    vector<int> beautifulArray(int n) {

        vector<int> res = {1};

        while(res.size() < n) {

            vector<int> temp;

            for(int num : res) {

                if(num \* 2 - 1 <= n) temp.push\_back(num\*2-1);

            }

            for(int num : res) {

                if(num\*2 <= n) temp.push\_back(num\*2);

            }

            res = temp;

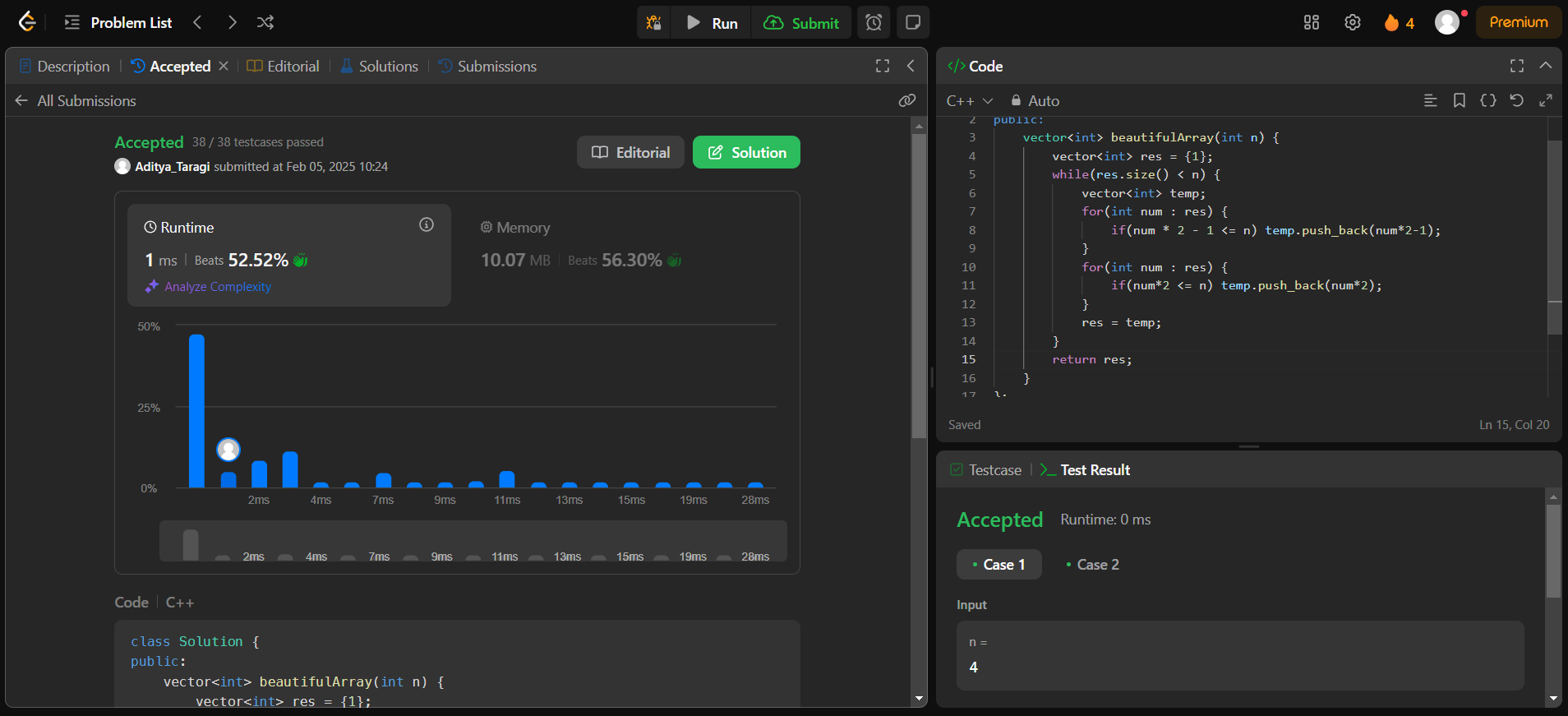
        }

        return res;

    }

};

**Screenshot:**



**Problem 6: Super Pow (**<https://leetcode.com/problems/super-pow/> **)**

**Code:**class Solution {

public:

    int modExp(int a, int b, int mod) {

        long long result = 1;

        a = a % mod;

        while(b>0) {

            if(b%2 == 1) {

                result = (result\*a) % mod;

            }

            a = (a\*a) % mod;

            b /=2;

        }

        return (int) result;

    }

    int superPow(int a, vector<int>& b) {

        int mod = 1337;

        a = a % mod;

        int result = 1;

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

            result = modExp(result,10,mod);

            result = (result \* modExp(a,b[i],mod)) % mod;

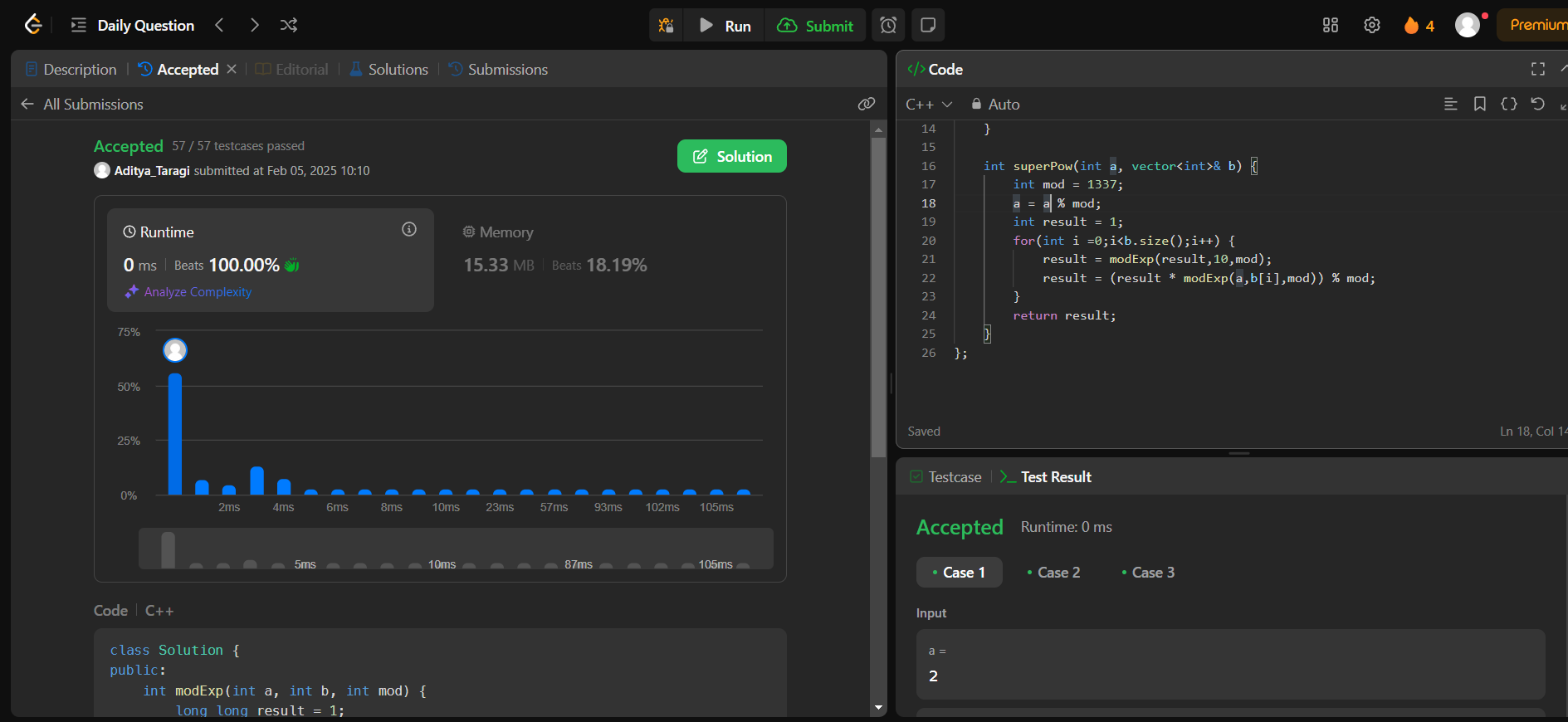
        }

        return result;

    }

};

**Screenshot:**



**Problem 7: The Skyline Problem (**<https://leetcode.com/problems/the-skyline-problem/> **)**

**Code:**class Solution {

public:

    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {

        vector<pair<int,int>> events;

        for(auto& b: buildings) {

            events.push\_back({b[0], -b[2]});

            events.push\_back({b[1],b[2]});

        }

        sort(events.begin(),events.end());

        multiset<int> heights = {0};

        vector<vector<int>> result;

        int prevHeight = 0;

        for(auto& [x,h] : events) {

            if(h<0) heights.insert(-h);

            else heights.erase(heights.find(h));

            int maxHeight = \*heights.rbegin();

            if(maxHeight != prevHeight) {

                result.push\_back({x, maxHeight});

                prevHeight = maxHeight;

            }

        }

        return result;

    }

};

**Screenshot:**

