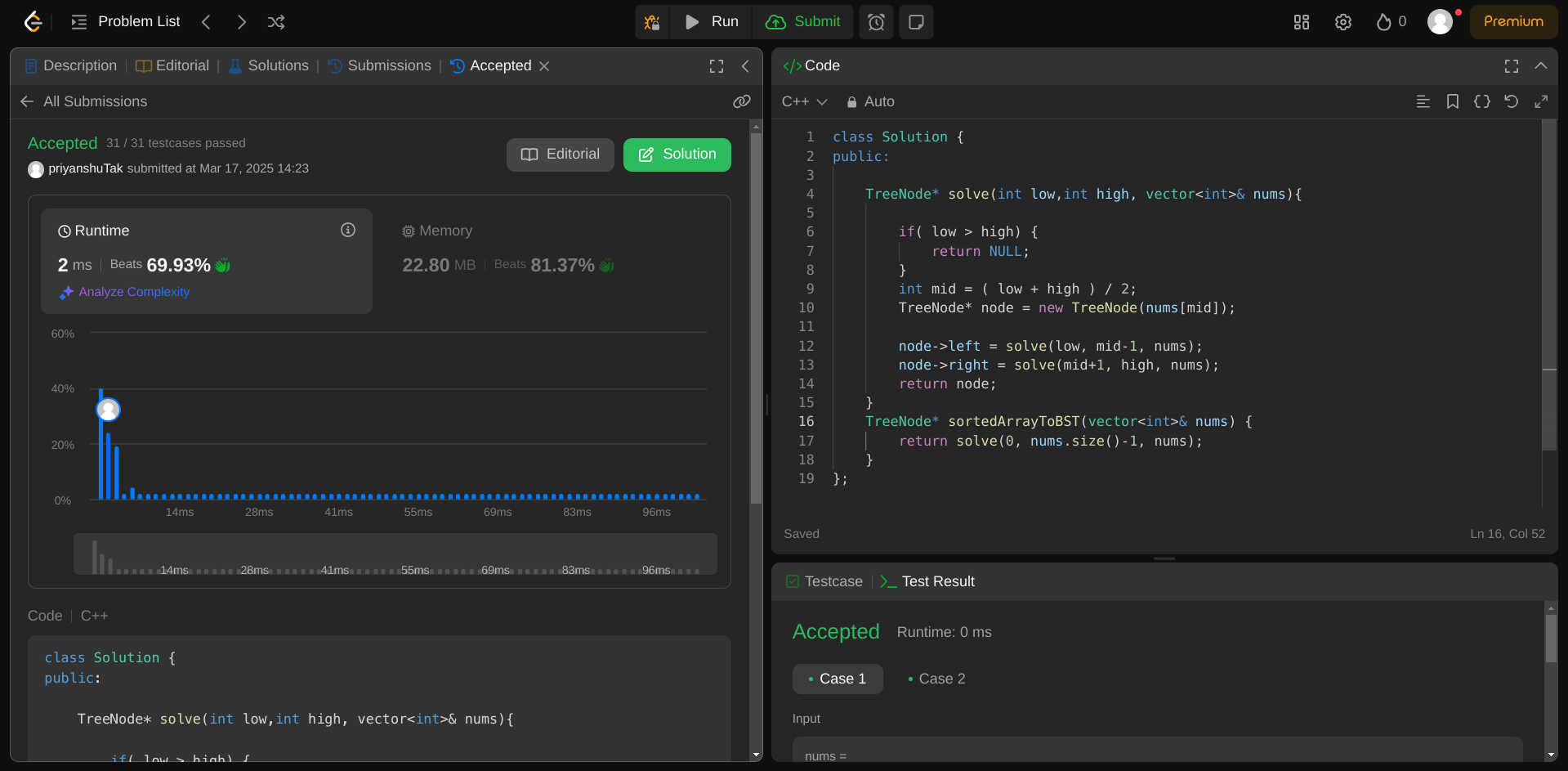
108.Convert Sorted Array to Binary Search Tree



class Solution {

public:

TreeNode\* solve(int low,int high, vector<int>& nums){

if( low > high) {

return NULL;

}

int mid = ( low + high ) / 2;

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

node->left = solve(low, mid-1, nums);

node->right = solve(mid+1, high, nums);

return node;

}

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

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

}

};

191.Number of 1 Bits

class Solution {

public:

int hammingWeight(uint32\_t n) {

int count = 0;

while(n != 0){

count += (n%2);

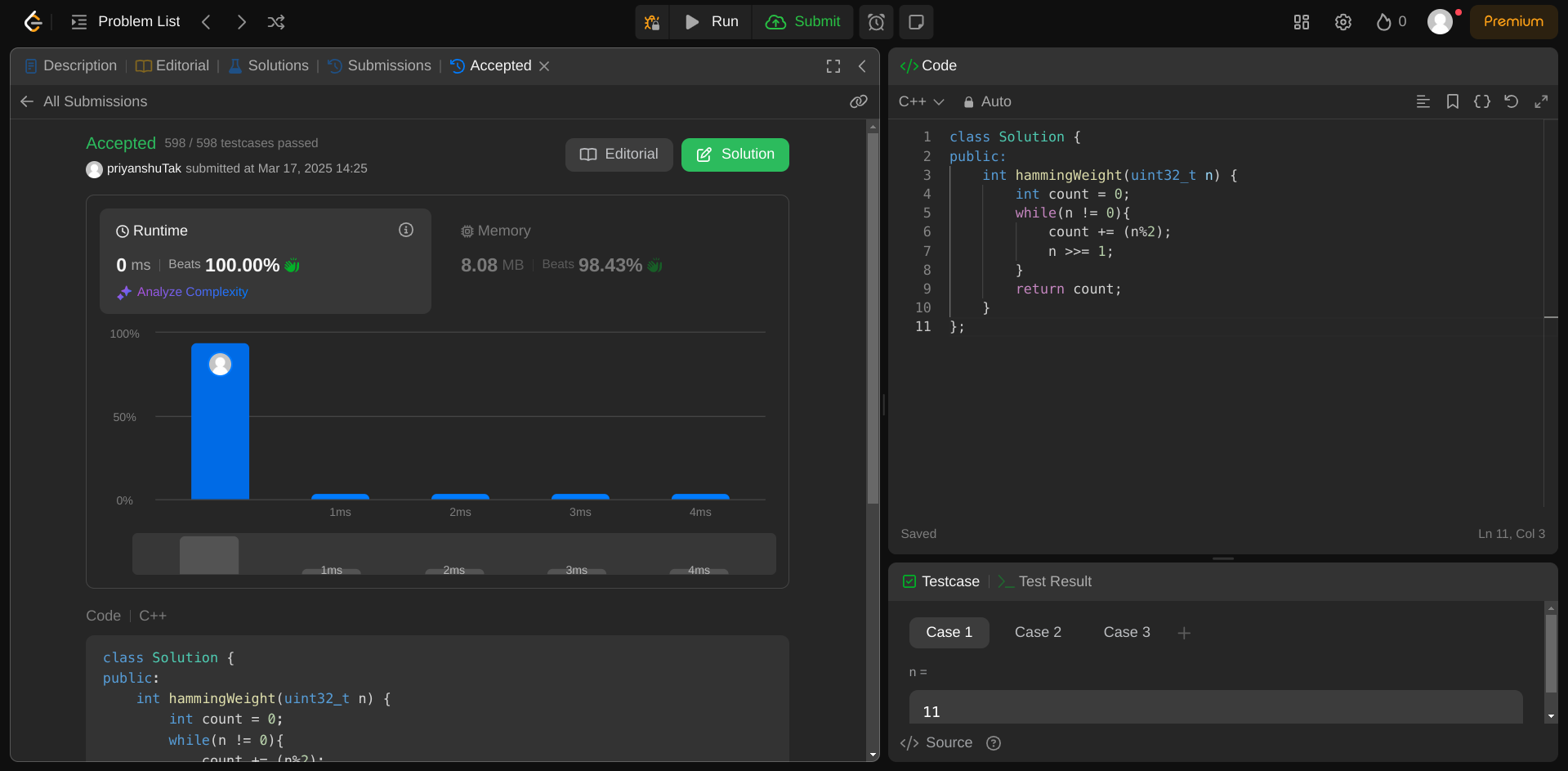
n >>= 1;

}

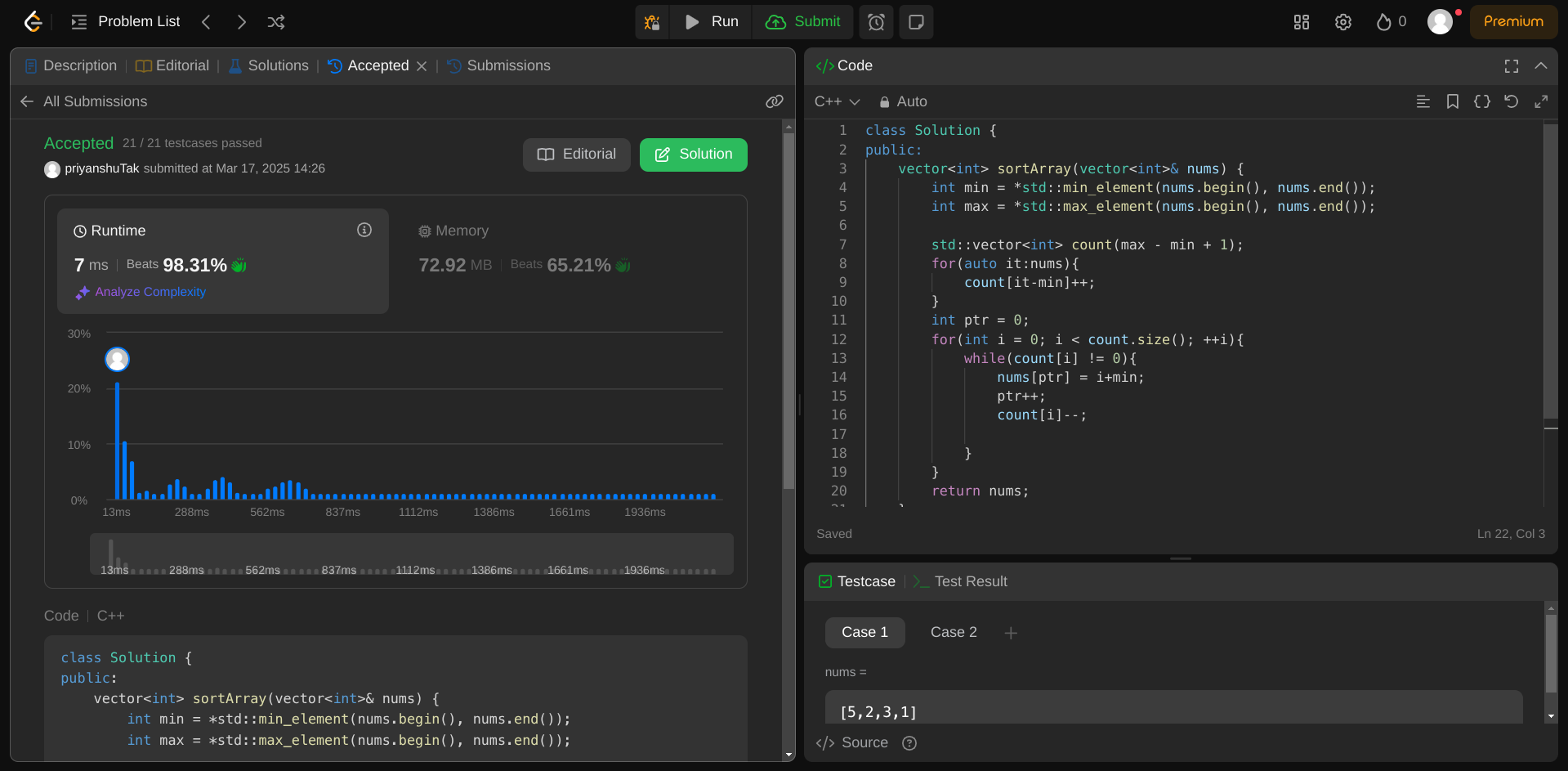
return count;

}

};



912.Sort an Array



class Solution {

public:

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

int min = \*std::min\_element(nums.begin(), nums.end());

int max = \*std::max\_element(nums.begin(), nums.end());

std::vector<int> count(max - min + 1);

for(auto it:nums){

count[it-min]++;

}

int ptr = 0;

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

while(count[i] != 0){

nums[ptr] = i+min;

ptr++;

count[i]--;

}

}

return nums;

}

};

53.Maximum Subarray

class Solution {

public:

int maxSubArray(vector<int>& nums) {

int currsum = nums[0];

int finalsum = nums[0];

for(int i=1;i<nums.size();i++){

if(currsum < 0){

currsum =nums[i];

}else{

currsum+=nums[i];

}

if(finalsum<currsum){

finalsum=currsum;

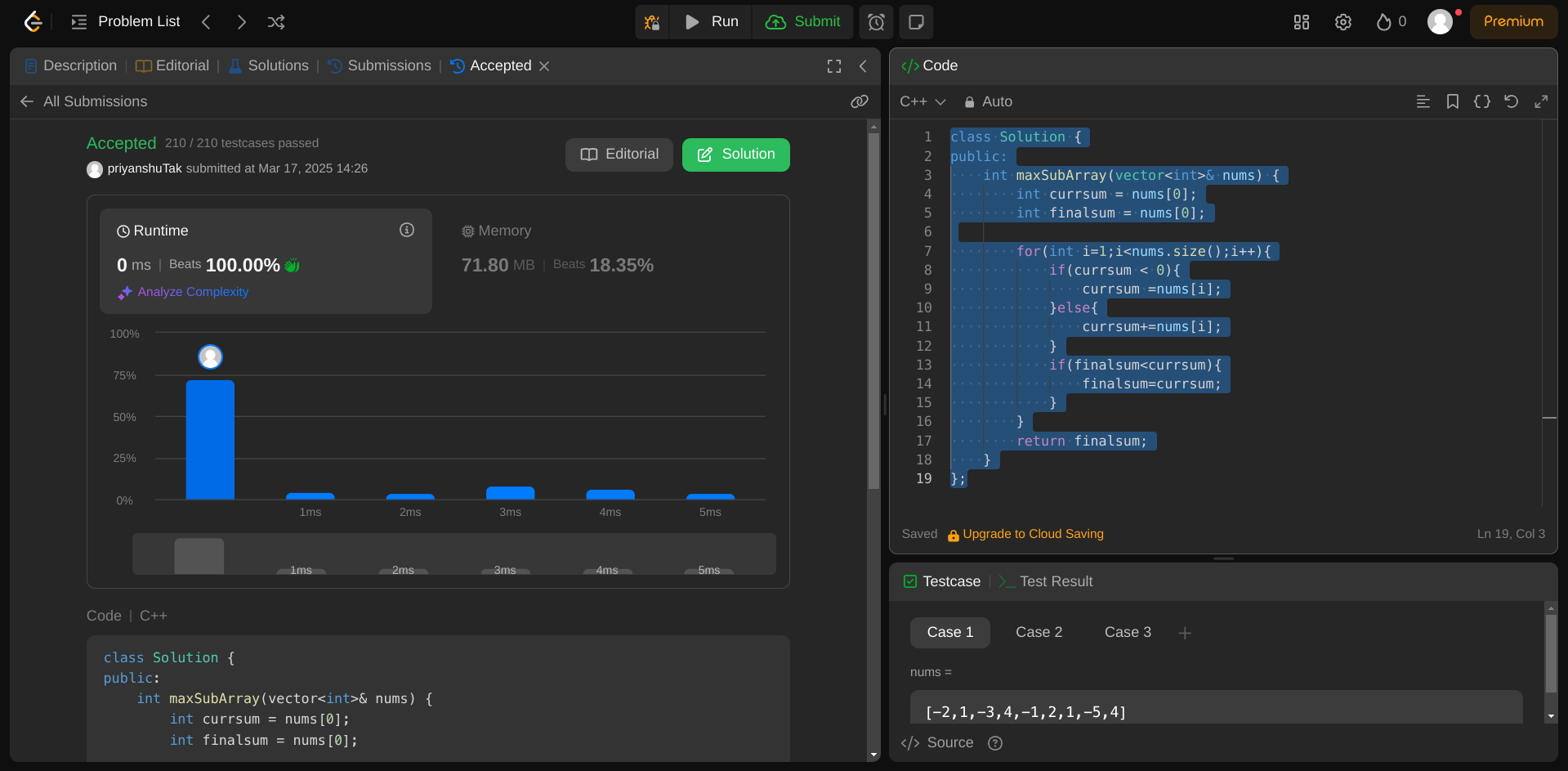
}

}

return finalsum;

}

};



932.Beautiful Array

class Solution {

public:

int partition(vector<int> &v, int start, int end, int mask)

{

int j = start;

for(int i = start; i <= end; i++)

{

if((v[i] & mask) != 0)

{

swap(v[i], v[j]);

j++;

}

}

return j;

}

void sort(vector<int> & v, int start, int end, int mask)

{

if(start >= end) return;

int mid = partition(v, start, end, mask);

sort(v, start, mid - 1, mask << 1);

sort(v, mid, end, mask << 1);

}

vector<int> beautifulArray(int N) {

vector<int> ans;

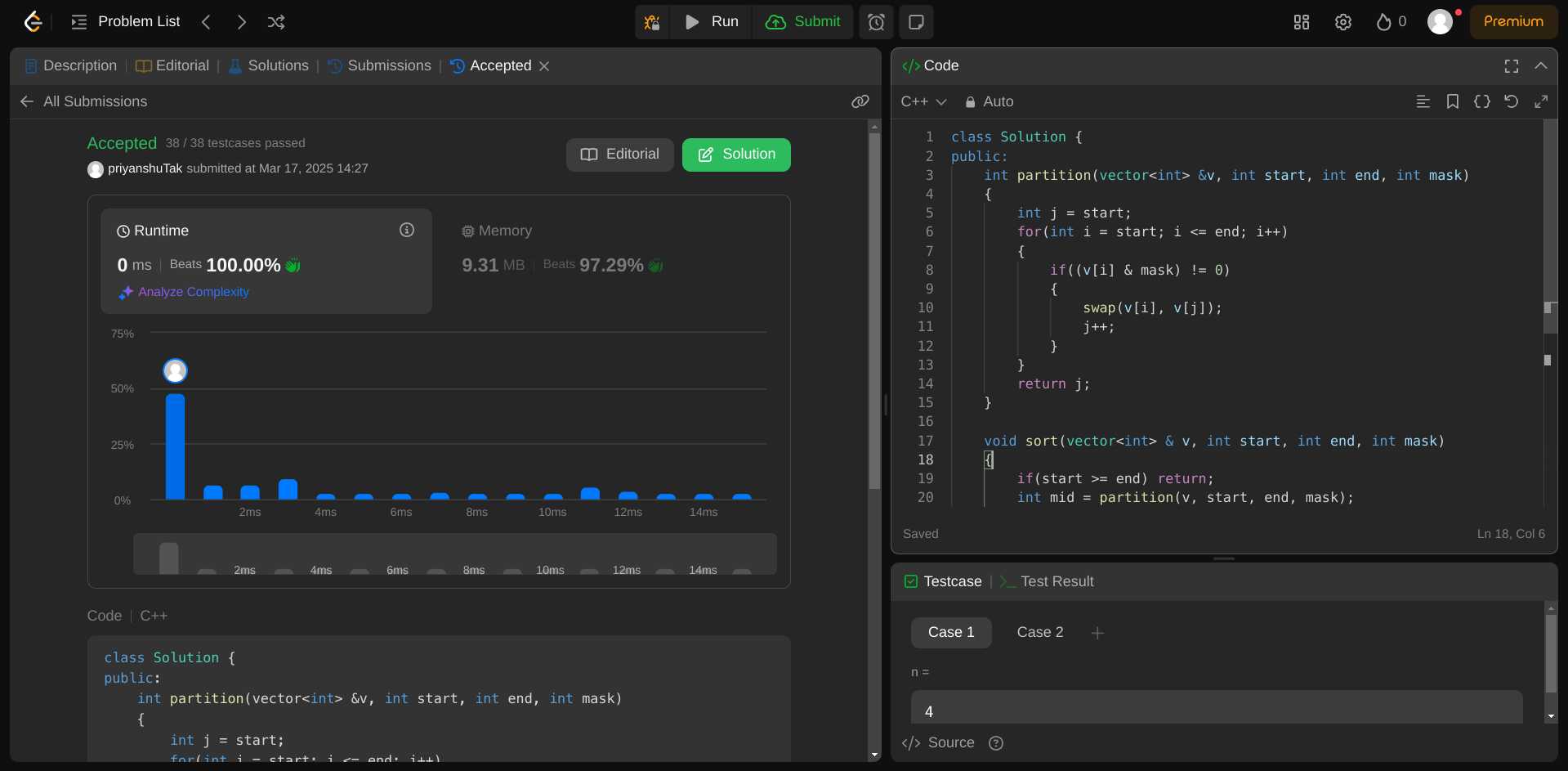
for(int i = 0; i < N; i++) ans.push\_back(i + 1);

sort(ans, 0, N - 1, 1);

return ans;

}

};



372.Super Pow

class Solution {

private:

int solve(int base, int power, int mod) {

int ans = 1;

while (power > 0) {

if (power & 1) {

ans = (ans \* base) % mod;

}

base = (base \* base) % mod;

power >>= 1;

}

return ans;

}

public:

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

a%=1337;

int n = b.size();

int m = 1140;

int expi = 0;

for(int i : b){

expi = (expi\*10+i)%m;

}

if (expi == 0) {

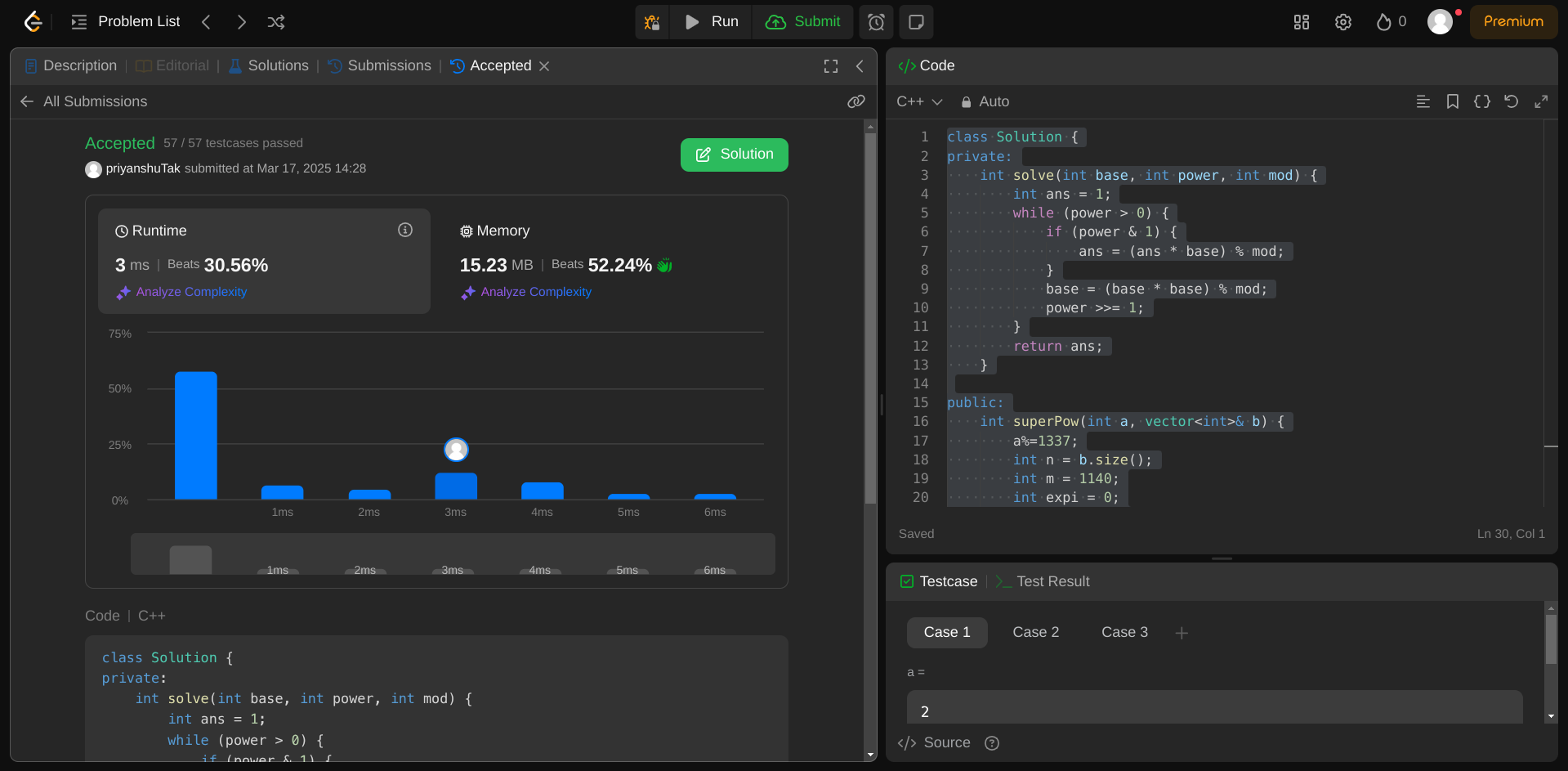
expi = m;

}

return solve(a,expi,1337);

}

};



218.The Skyline Problem

class Solution {

public:

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

vector<vector<int>> ans;

multiset<int> pq{0};

vector<pair<int, int>> points;

for(auto b: buildings){

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

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

}

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

int ongoingHeight = 0;

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

int currentPoint = points[i].first;

int heightAtCurrentPoint = points[i].second;

if(heightAtCurrentPoint < 0){

pq.insert(-heightAtCurrentPoint);

} else {

pq.erase(pq.find(heightAtCurrentPoint));

}

auto pqTop = \*pq.rbegin();

if(ongoingHeight != pqTop){

ongoingHeight = pqTop;

ans.push\_back({currentPoint, ongoingHeight});

}

}

return ans;

}

};

