[**1763. Longest Nice Substring**](https://leetcode.com/problems/longest-nice-substring/)

**Code**

class Solution {

public:

bool check(string s){

vector<int> t1(26, 0), t2(26, 0);

for(int i=0; i<s.length(); i++) if(s[i] >= 'a' && s[i] <= 'z') t1[s[i] - 'a']++;

for(int i=0; i<s.length(); i++) if(s[i] >= 'A' && s[i] <= 'Z') t2[s[i] - 'A']++;

for(int i=0; i<26; i++){

if(t1[i] == 0 && t2[i] == 0) continue;

else if(t1[i] == 0 && t2[i] > 0) return false;

else if(t1[i] > 0 && t2[i] == 0) return false;

}

return true;

}

string longestNiceSubstring(string s) {

if(s == "") return "";

if(check(s)) return s;

unordered\_map<char,int> m;

vector<int> t1(26, 0), t2(26, 0);

for(int i=0; i<s.length(); i++) if(s[i] >= 'a' && s[i] <= 'z') t1[s[i] - 'a']++;

for(int i=0; i<s.length(); i++) if(s[i] >= 'A' && s[i] <= 'Z') t2[s[i] - 'A']++;

for(int i=0; i<26; i++){

if(t1[i] == 0 && t2[i] == 0) continue;

else if(t1[i] == 0 && t2[i] > 0) m[i + 'A']++;

else if(t1[i] > 0 && t2[i] == 0) m[i + 'a']++;

}

vector<string> ans; string temp = "";

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

if(m.find(s[i]) == m.end()) temp += s[i];

else {

ans.push\_back(temp); temp = "";

}

}

if(temp != "") ans.push\_back(temp);

string kk = "";

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

if(ans[i] == "") continue;

string t = longestNiceSubstring(ans[i]);

if(t == "") continue;

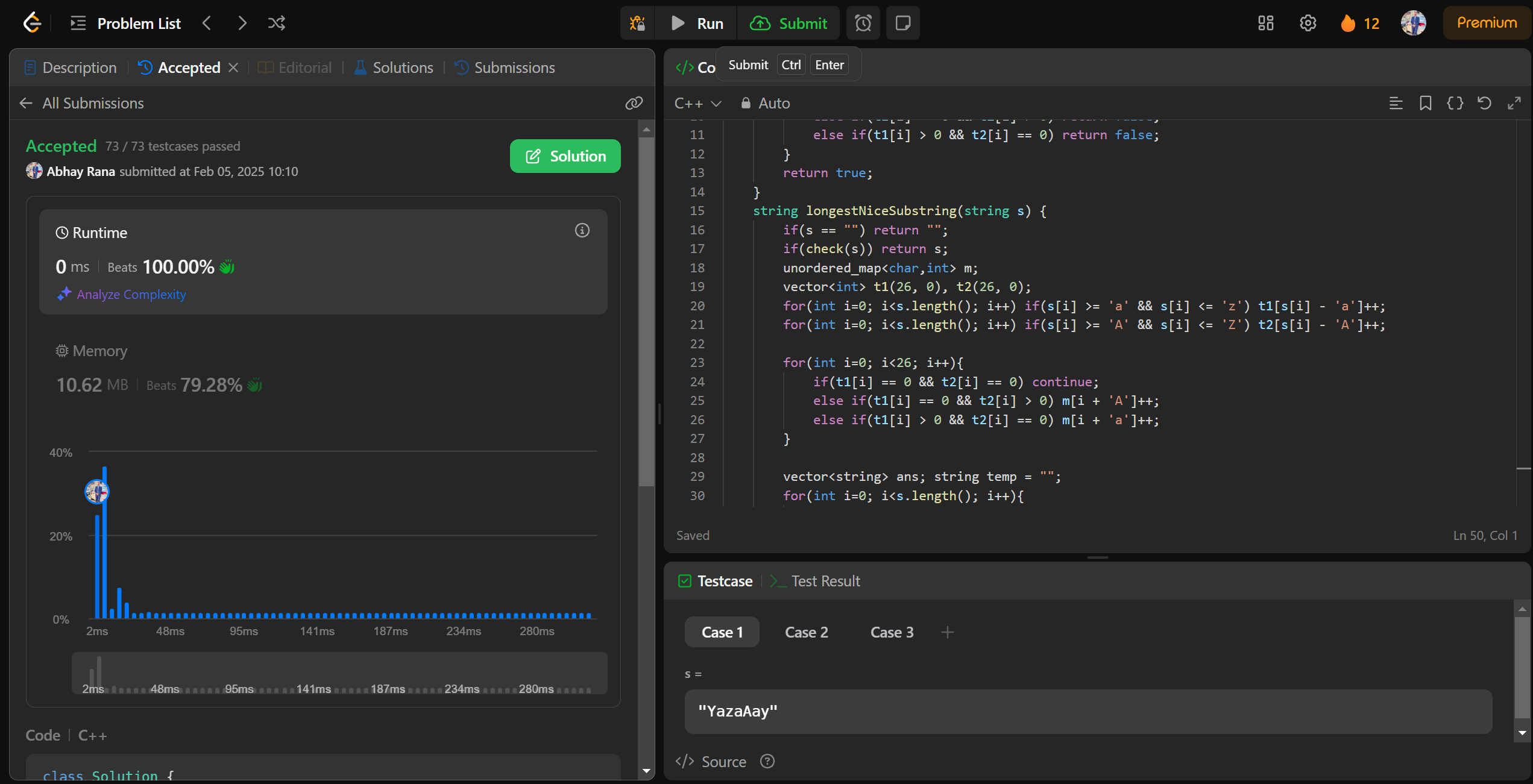
if(kk == "" || kk.length() < t.length()) kk = t;

}

return kk;

}

};



[**190. Reverse Bits**](https://leetcode.com/problems/reverse-bits/)

class Solution {

public:

uint32\_t reverseBits(uint32\_t n) {

uint32\_t result = 0;

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

int bit = n & 1; // Extract the least significant bit

result = (result << 1) | bit; // Append the bit to the result

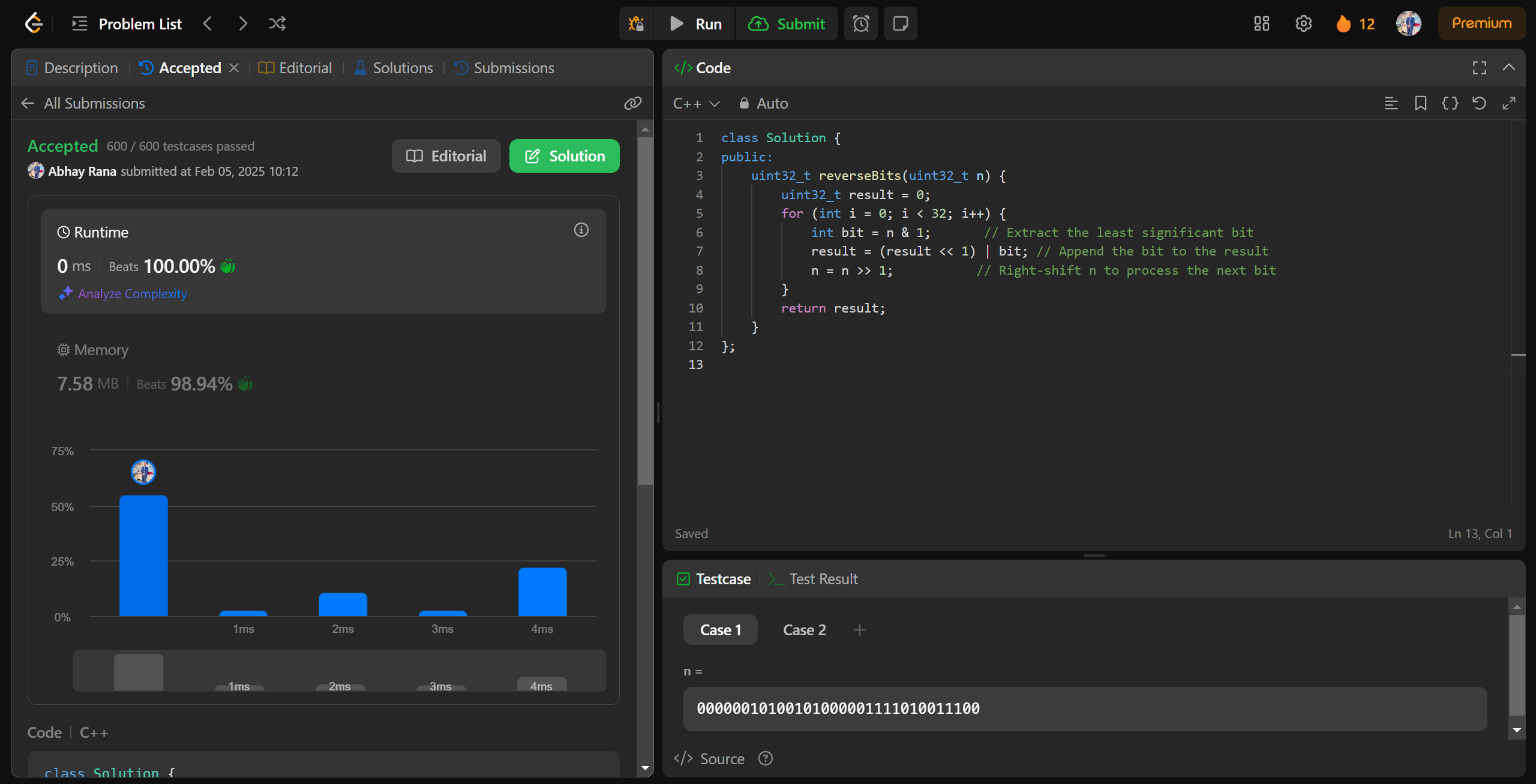
n = n >> 1; // Right-shift n to process the next bit

}

return result;

}

};



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

class Solution {

public:

int hammingWeight(uint32\_t n) {

int count = 0;

while(n != 0){

count += (n%2);

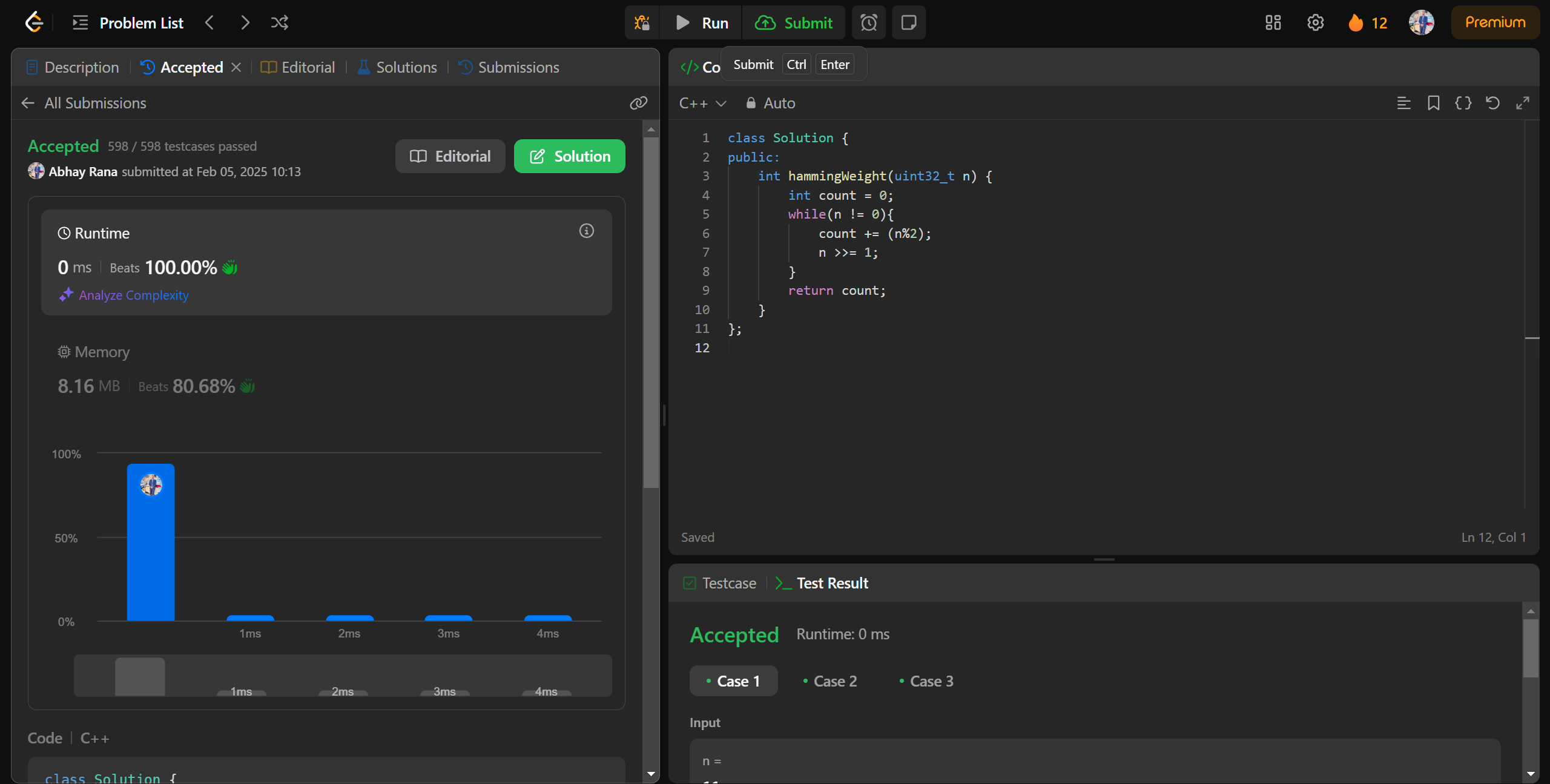
n >>= 1;

}

return count;

}

};



[**53. Maximum Subarray**](https://leetcode.com/problems/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;

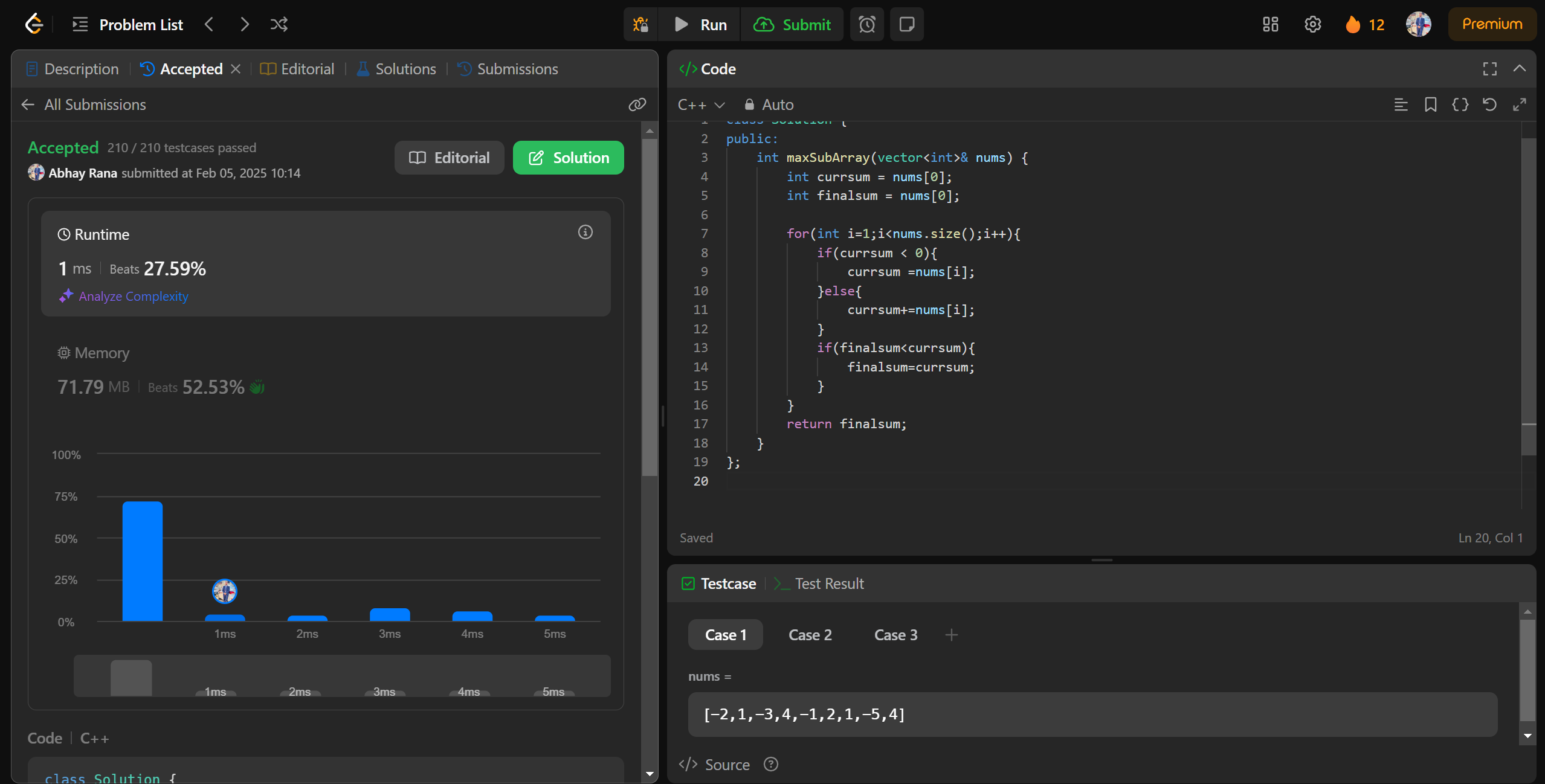
}

}

return finalsum;

}

};



[**240. Search a 2D Matrix II**](https://leetcode.com/problems/search-a-2d-matrix-ii/)

class Solution {

public:

    bool searchMatrix(vector<vector<int>>& matrix, int target) {

        int n=matrix.size(), m=matrix[0].size();

        int row=n-1, col=0;

        while(col < m && row >= 0)

        {

            if(matrix[row][col]==target) return true;

            else if(matrix[row][col] < target) col++;

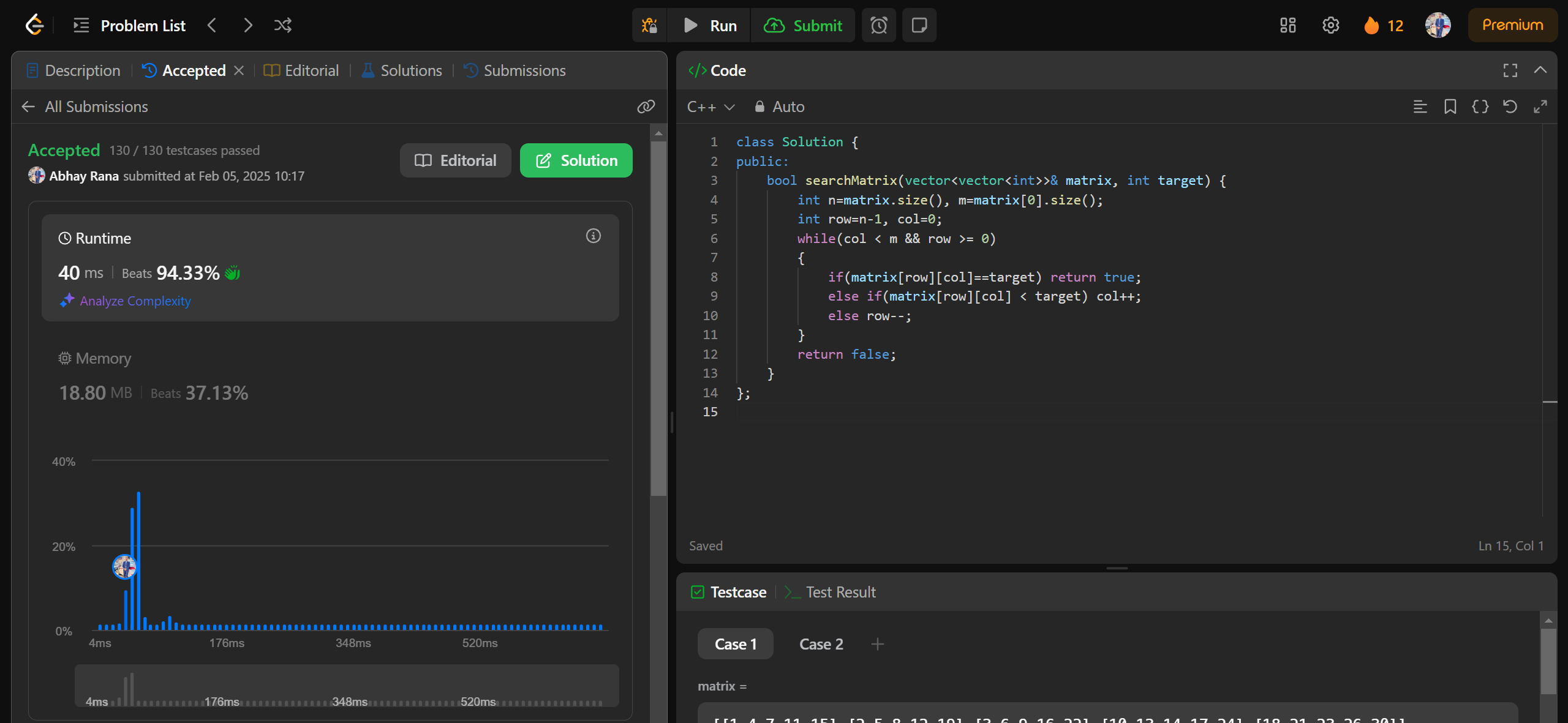
            else row--;

        }

        return false;

    }

};



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

class Solution {

    const int base = 1337;

    int powmod(int a, int k) //a^k mod 1337 where 0 <= k <= 10

    {

        a %= base;

        int result = 1;

        for (int i = 0; i < k; ++i)

            result = (result \* a) % base;

        return result;

    }

public:

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

        if (b.empty()) return 1;

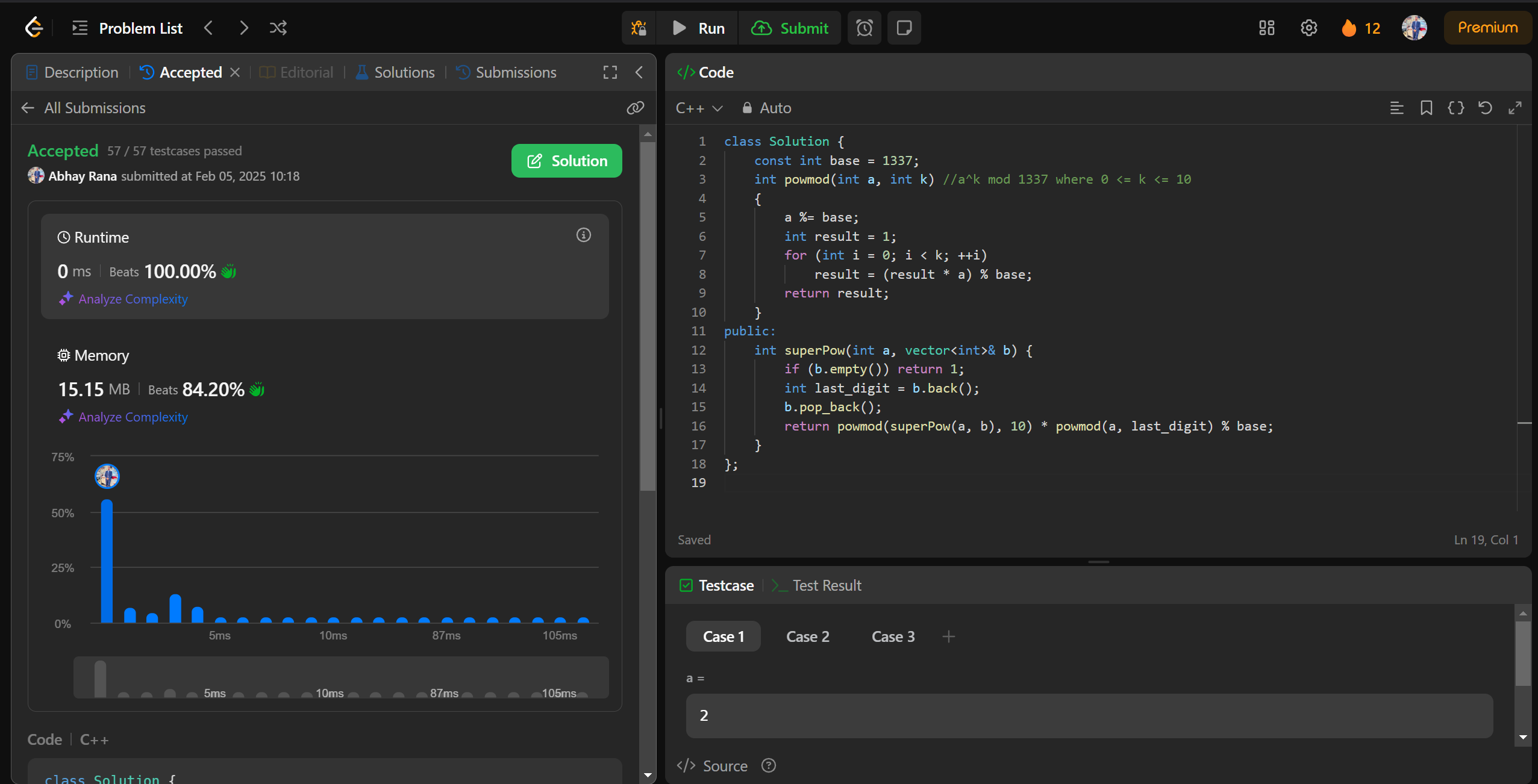
        int last\_digit = b.back();

        b.pop\_back();

        return powmod(superPow(a, b), 10) \* powmod(a, last\_digit) % base;

    }

};



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

class Solution {

public:

    vector<int> beautifulArray(int N) {

        vector<int> res = {1};

        while (res.size() < N) {

            vector<int> tmp;

            for (int i : res) if (i \* 2 - 1 <= N) tmp.push\_back(i \* 2 - 1);

            for (int i : res) if (i \* 2 <= N) tmp.push\_back(i \* 2);

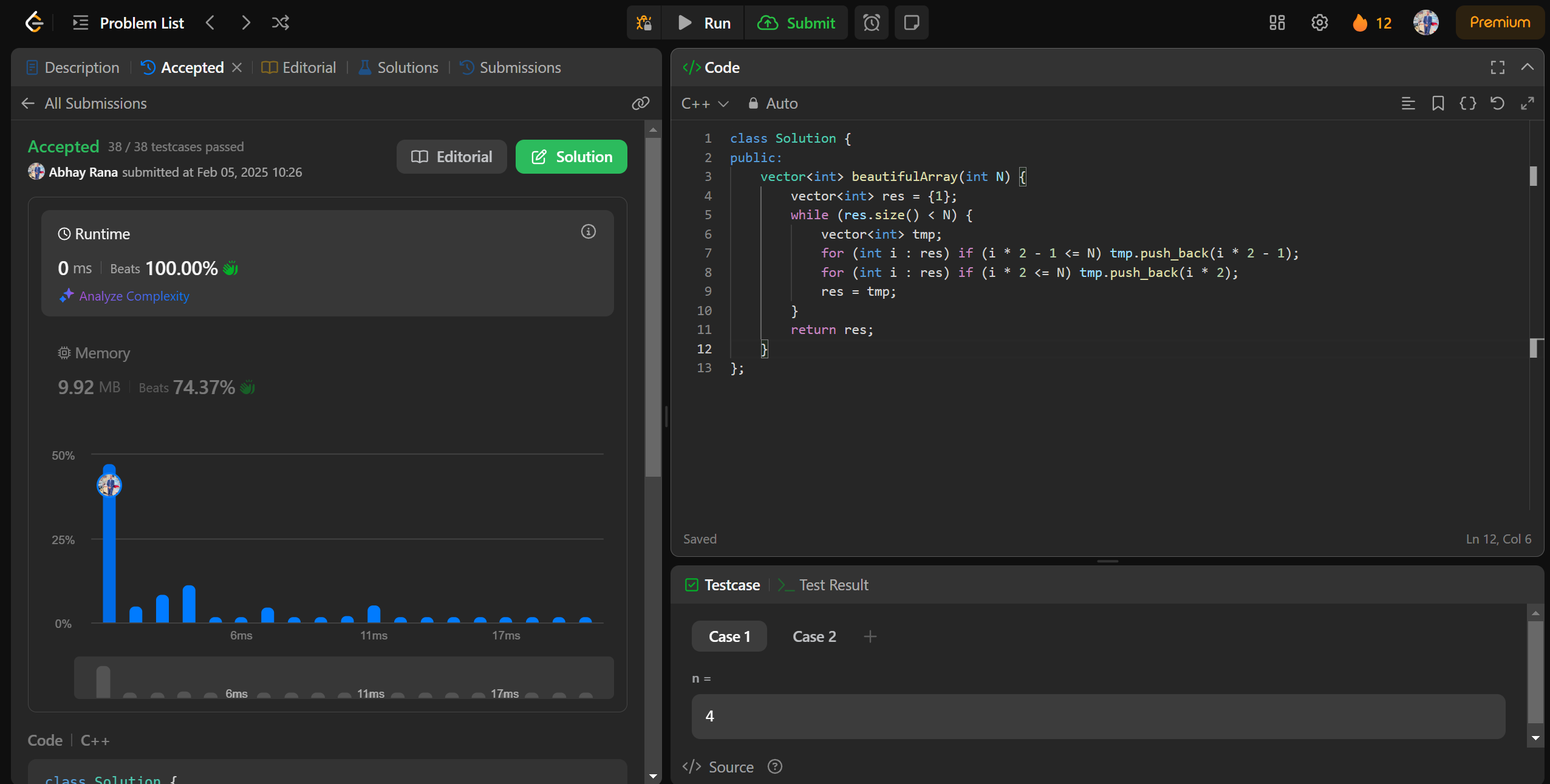
            res = tmp;

        }

        return res;

    }

};



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

class Solution {

public:

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

int edge\_idx = 0;

vector<pair<int, int>> edges;

priority\_queue<pair<int, int>> pq;

vector<vector<int>> skyline;

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

const auto &b = buildings[i];

edges.emplace\_back(b[0], i);

edges.emplace\_back(b[1], i);

}

std::sort(edges.begin(), edges.end());

while (edge\_idx < edges.size()) {

int curr\_height;

const auto &[curr\_x, \_] = edges[edge\_idx];

while (edge\_idx < edges.size() &&

curr\_x == edges[edge\_idx].first) {

const auto &[\_, building\_idx] = edges[edge\_idx];

const auto &b = buildings[building\_idx];

if (b[0] == curr\_x)

pq.emplace(b[2], b[1]);

++edge\_idx;

}

while (!pq.empty() && pq.top().second <= curr\_x)

pq.pop();

curr\_height = pq.empty() ? 0 : pq.top().first;

if (skyline.empty() || skyline.back()[1] != curr\_height)

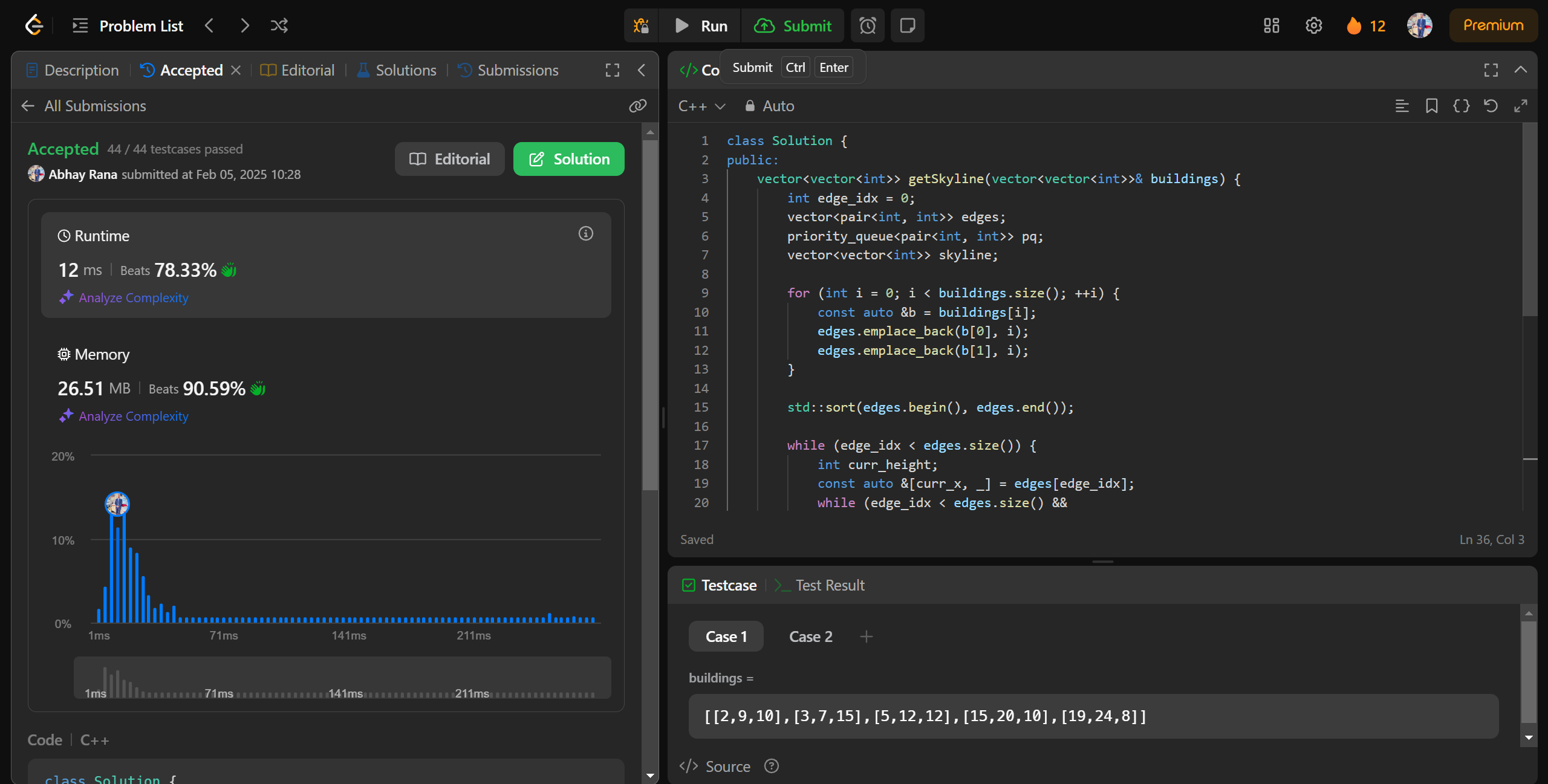
skyline.push\_back({curr\_x, curr\_height});

}

return skyline;

}

};



493.[Reverse Pairs](https://leetcode.com/problems/reverse-pairs/description/)

class SegTree {

private:

int tree\_size;

vector<int> tree;

void update(int lx, int rx, int ni, int idx) {

if (rx - lx == 1) {

tree[ni]++;

return;

}

int m = (lx + rx) >> 1;

if (idx < m)

update(lx, m, ni \* 2 + 1, idx);

else

update(m, rx, ni \* 2 + 2, idx);

tree[ni] = tree[ni \* 2 + 1] + tree[ni \* 2 + 2];

}

int query(int l, int r, int lx, int rx, int ni) {

if (l >= rx || r <= lx)

return 0;

if (l <= lx && r >= rx)

return tree[ni];

int m = (lx + rx) >> 1;

return query(l, r, lx, m, ni \* 2 + 1) + query(l, r, m, rx, ni \* 2 + 2);

}

public:

SegTree(int n) {

tree\_size = 1;

while (tree\_size < n)

tree\_size <<= 1;

tree = vector<int>(tree\_size \* 2);

}

void update(int idx) {

update(0, tree\_size, 0, idx);

}

int query(int l, int r) {

return query(l, r + 1, 0, tree\_size, 0);

}

};

class Solution {

public:

int reversePairs(vector<int>& nums) {

int n = nums.size();

set<long long> values;

for(const auto& num : nums) {

values.insert(num);

values.insert(2LL \* num);

}

int last\_index = 0;

unordered\_map<long long, int> values\_indices;

for(const auto& val : values)

values\_indices[val] = last\_index++;

SegTree seg\_tree(last\_index);

int ans = 0;

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

ans += seg\_tree.query(values\_indices[2LL \* nums[i]] + 1, last\_index);

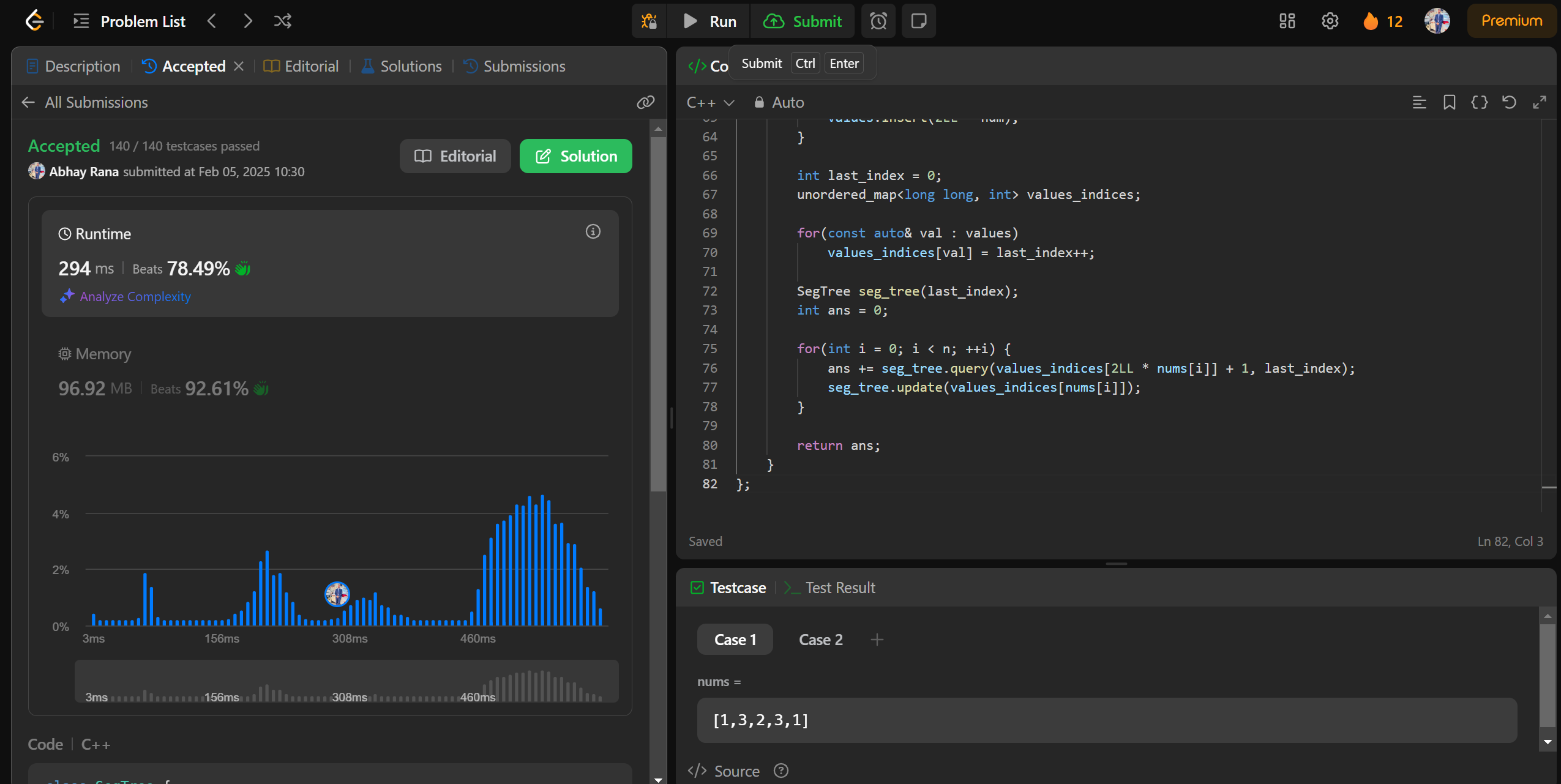
seg\_tree.update(values\_indices[nums[i]]);

}

return ans;

}

};



2407.[Longest Increasing Subsequence II](https://leetcode.com/problems/longest-increasing-subsequence-ii/description/)

class MaxSegmentTree {

public:

int n;

vector<int> tree;

MaxSegmentTree(int n\_) : n(n\_) {

int size = (int)(ceil(log2(n)));

size = (2 \* pow(2, size)) - 1;

tree = vector<int>(size);

}

int max\_value() { return tree[0]; }

int query(int l, int r) { return query\_util(0, l, r, 0, n - 1); }

int query\_util(int i, int qL, int qR, int l, int r) {

if (l >= qL && r <= qR) return tree[i];

if (l > qR || r < qL) return INT\_MIN;

int m = (l + r) / 2;

return max(query\_util(2 \* i + 1, qL, qR, l, m), query\_util(2 \* i + 2, qL, qR, m + 1, r));

}

void update(int i, int val) { update\_util(0, 0, n - 1, i, val); }

void update\_util(int i, int l, int r, int pos, int val) {

if (pos < l || pos > r) return;

if (l == r) {

tree[i] = max(val, tree[i]);

return;

}

int m = (l + r) / 2;

update\_util(2 \* i + 1, l, m, pos, val);

update\_util(2 \* i + 2, m + 1, r, pos, val);

tree[i] = max(tree[2 \* i + 1], tree[2 \* i + 2]);

}

};

class Solution {

public:

int lengthOfLIS(vector<int>& nums, int k) {

MaxSegmentTree tree(1e5 + 1);

for (int i : nums) {

int lower = max(0, i - k);

int cur = 1 + tree.query(lower, i - 1);

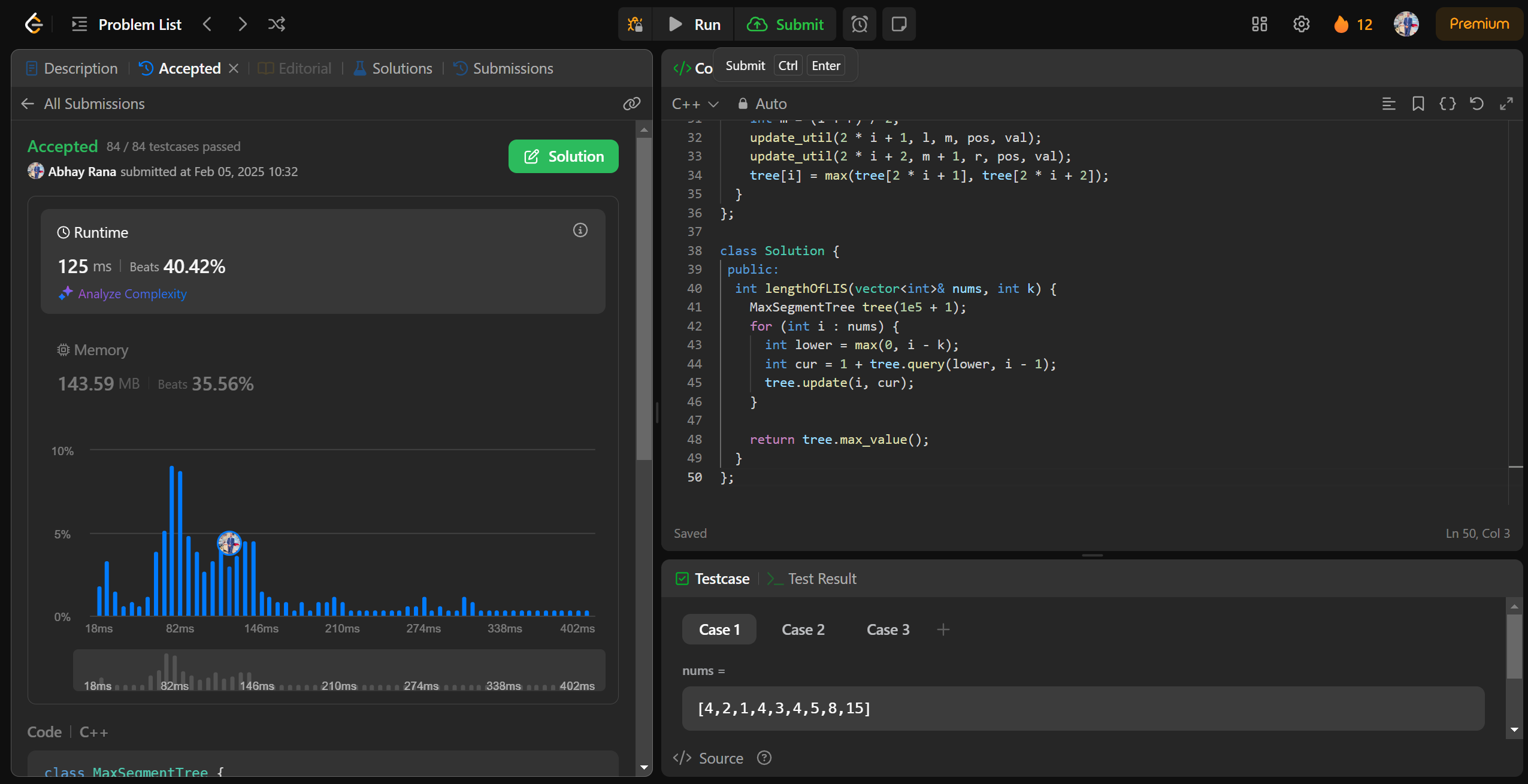
tree.update(i, cur);

}

return tree.max\_value();

}

};



88.[Merge Sorted Array](https://leetcode.com/problems/merge-sorted-array/description/)

class Solution {

public:

void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {

for (int j = 0, i = m; j<n; j++){

nums1[i] = nums2[j];

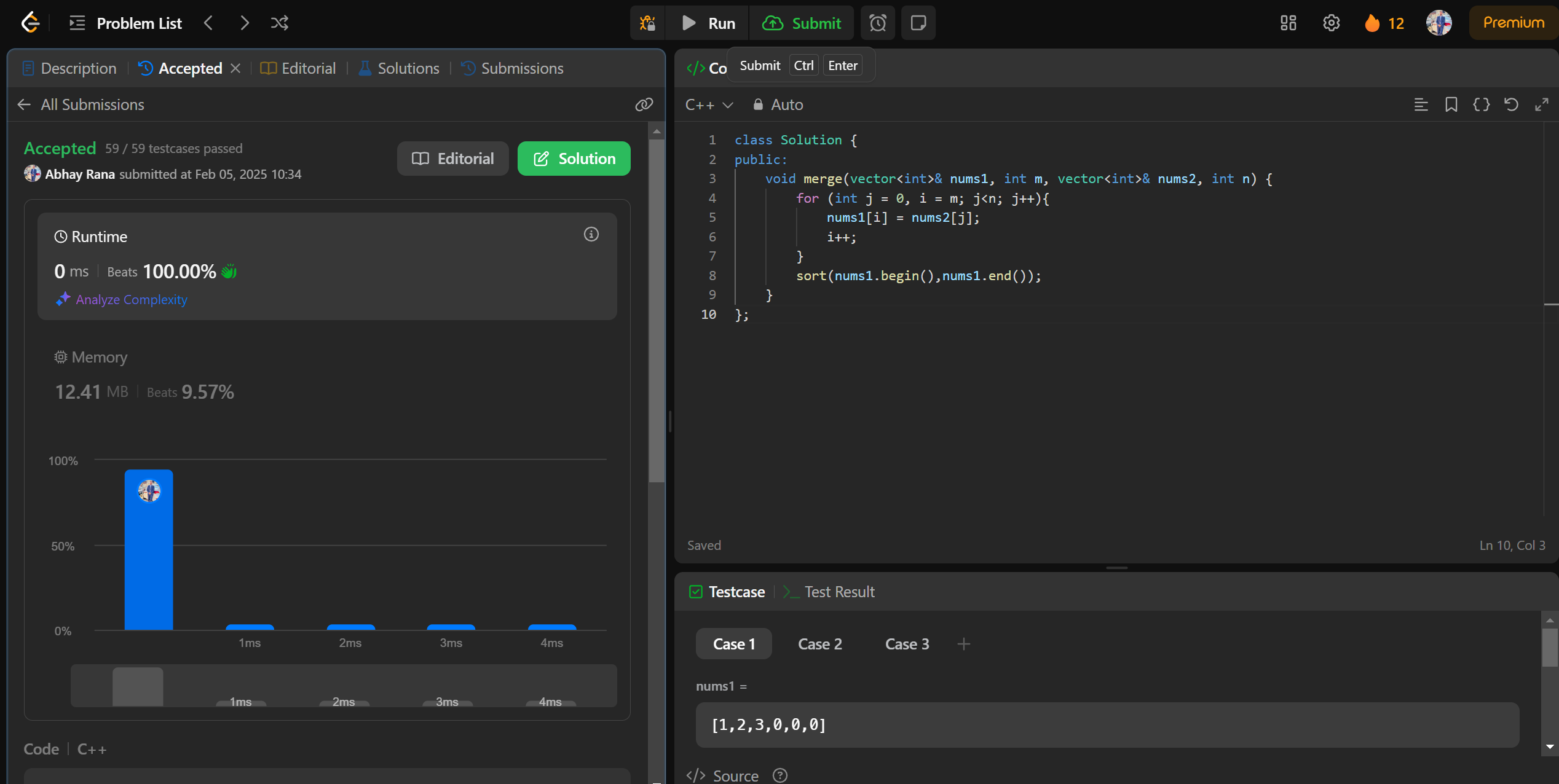
i++;

}

sort(nums1.begin(),nums1.end());

}

};



278.[First Bad Version](https://leetcode.com/problems/first-bad-version/description/)

class Solution {

public:

    int firstBadVersion(int n) {

        int low=1;

        int high=n;

        while(low<=high)

        {

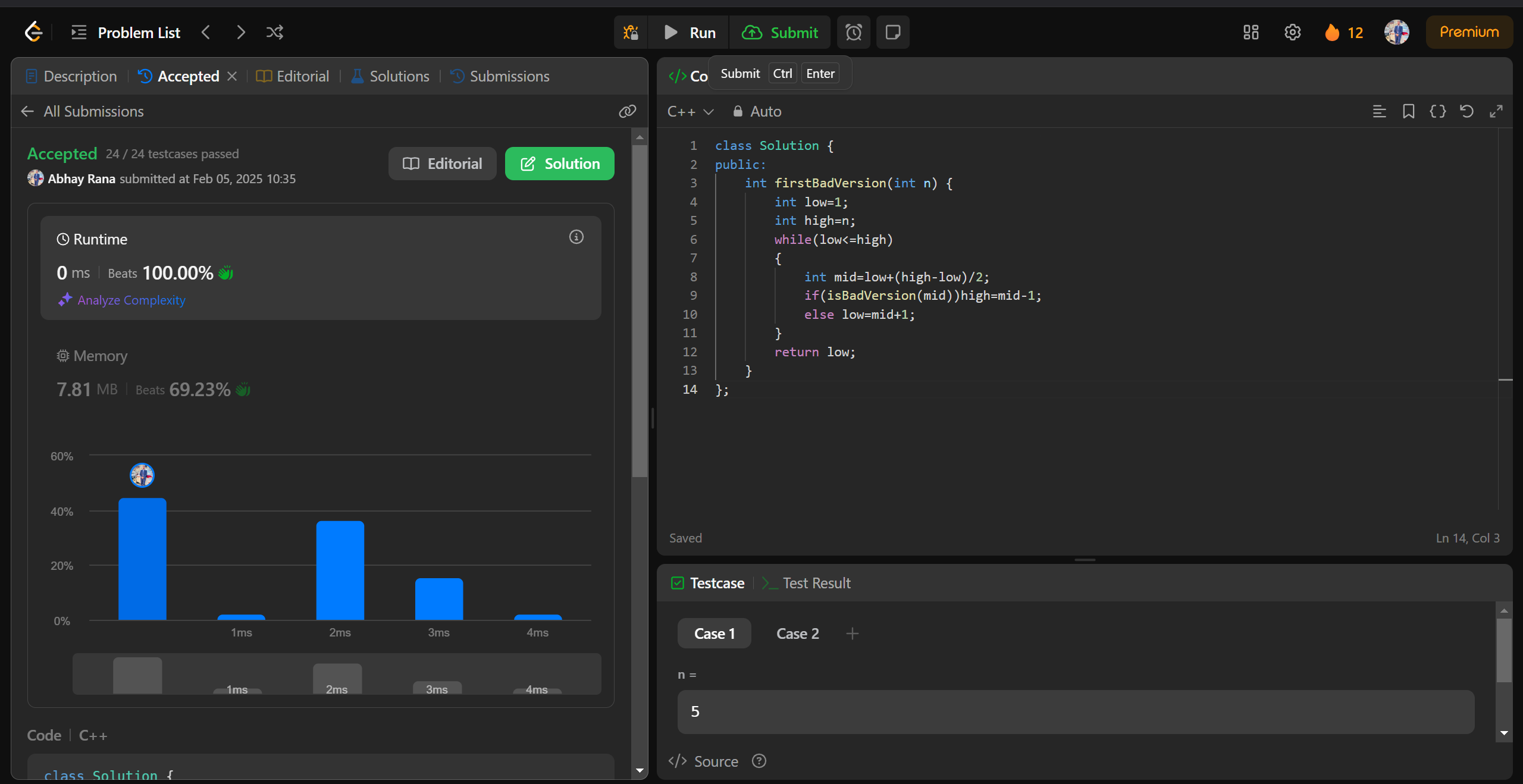
            int mid=low+(high-low)/2;

            if(isBadVersion(mid))high=mid-1;

            else low=mid+1;

        }

        return low;

    }};

75.[Sort Colors](https://leetcode.com/problems/sort-colors/description/)

class Solution {

public:

void sortColors(vector<int>& nums) {

int l=nums.size();

int count0=0;

int count1=0;

int count2=0;

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

if(nums[i]==0) count0++;

else if(nums[i]==1) count1++;

else count2++;

}

count1=count0+count1;

count2=count0+count1+count2;

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

if(i<count0) nums[i]=0;

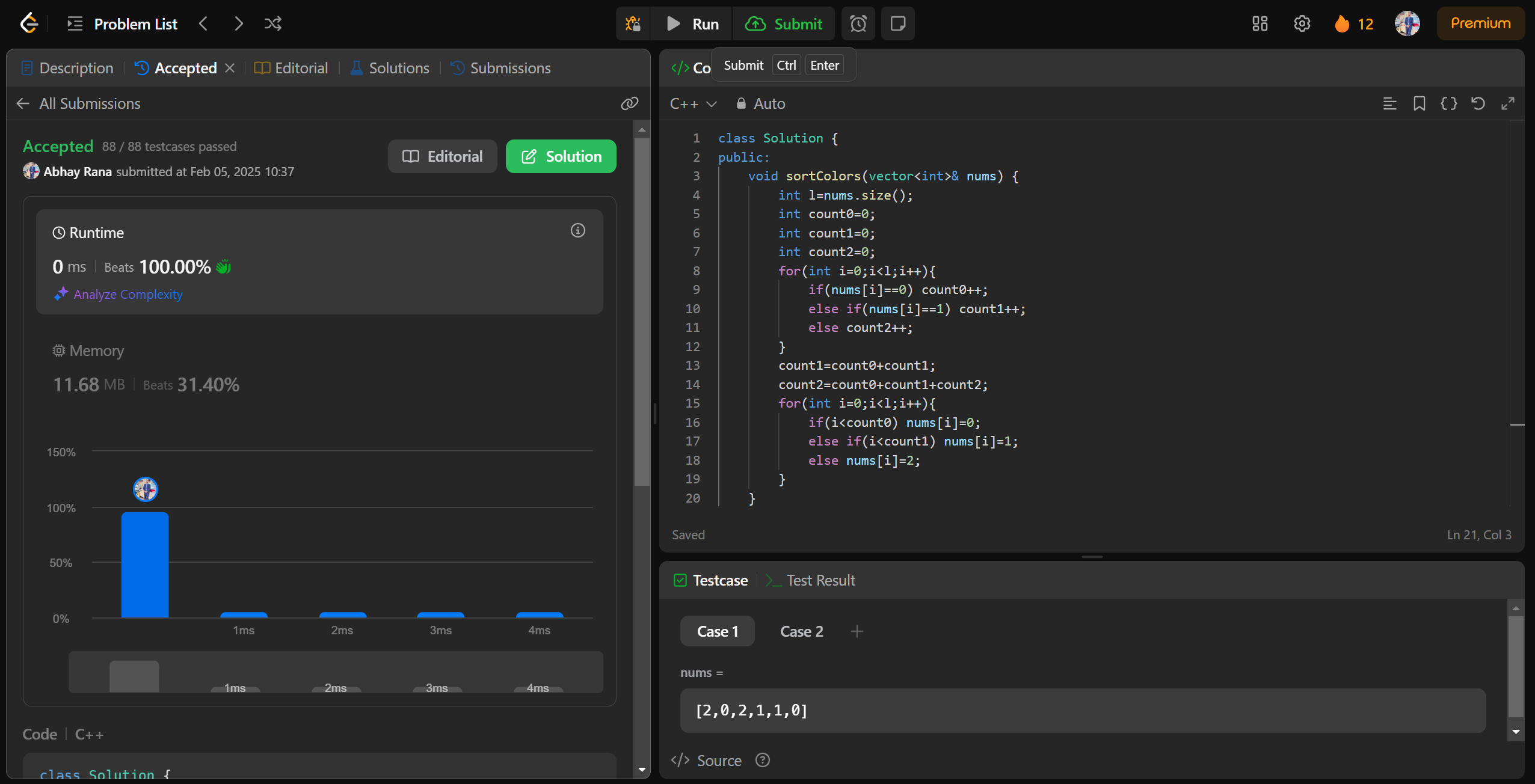
else if(i<count1) nums[i]=1;

else nums[i]=2;

}

}

};



347.[Top K Frequent Elements](https://leetcode.com/problems/top-k-frequent-elements/description/)

class Solution {

public:

vector<int> topKFrequent(vector<int>& nums, int k) {

unordered\_map<int, int> counter;

for (int n : nums) {

counter[n]++;

}

auto comp = [](pair<int, int>& a, pair<int, int>& b) {

return a.second < b.second;

};

priority\_queue<pair<int, int>, vector<pair<int, int>>, decltype(comp)> heap(comp);

for (auto& entry : counter) {

heap.push({entry.first, entry.second});

}

vector<int> res;

while (k-- > 0) {

res.push\_back(heap.top().first);

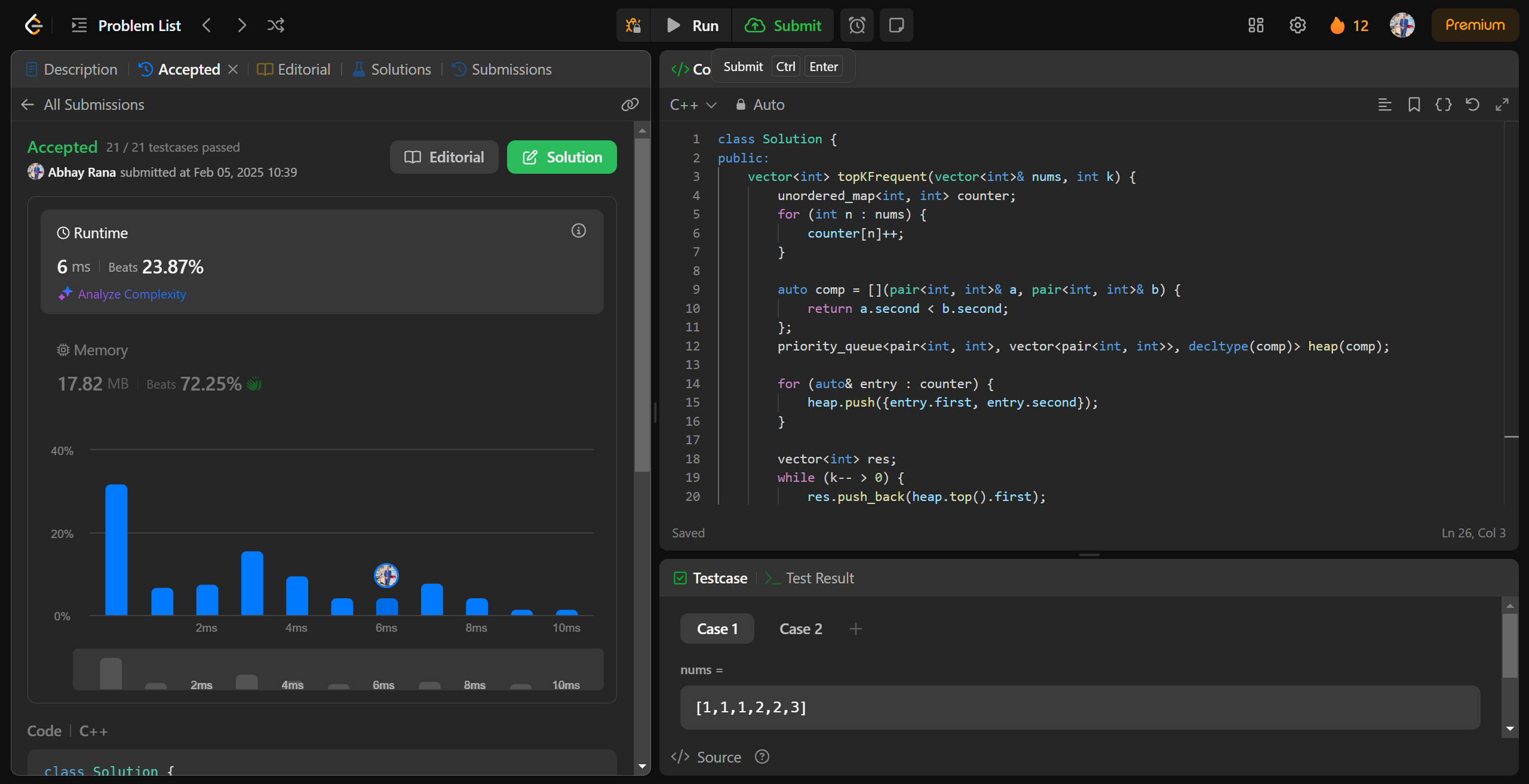
heap.pop();

}

return res;

}

};



215.[Kth Largest Element in an Array](https://leetcode.com/problems/kth-largest-element-in-an-array/description/)

class Solution {

public:

int findKthLargest(std::vector<int>& nums, int k) {

std::priority\_queue<int, std::vector<int>, std::greater<int>> min\_heap(nums.begin(), nums.begin() + k);

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

if (nums[i] > min\_heap.top()) {

min\_heap.pop();

min\_heap.push(nums[i]);

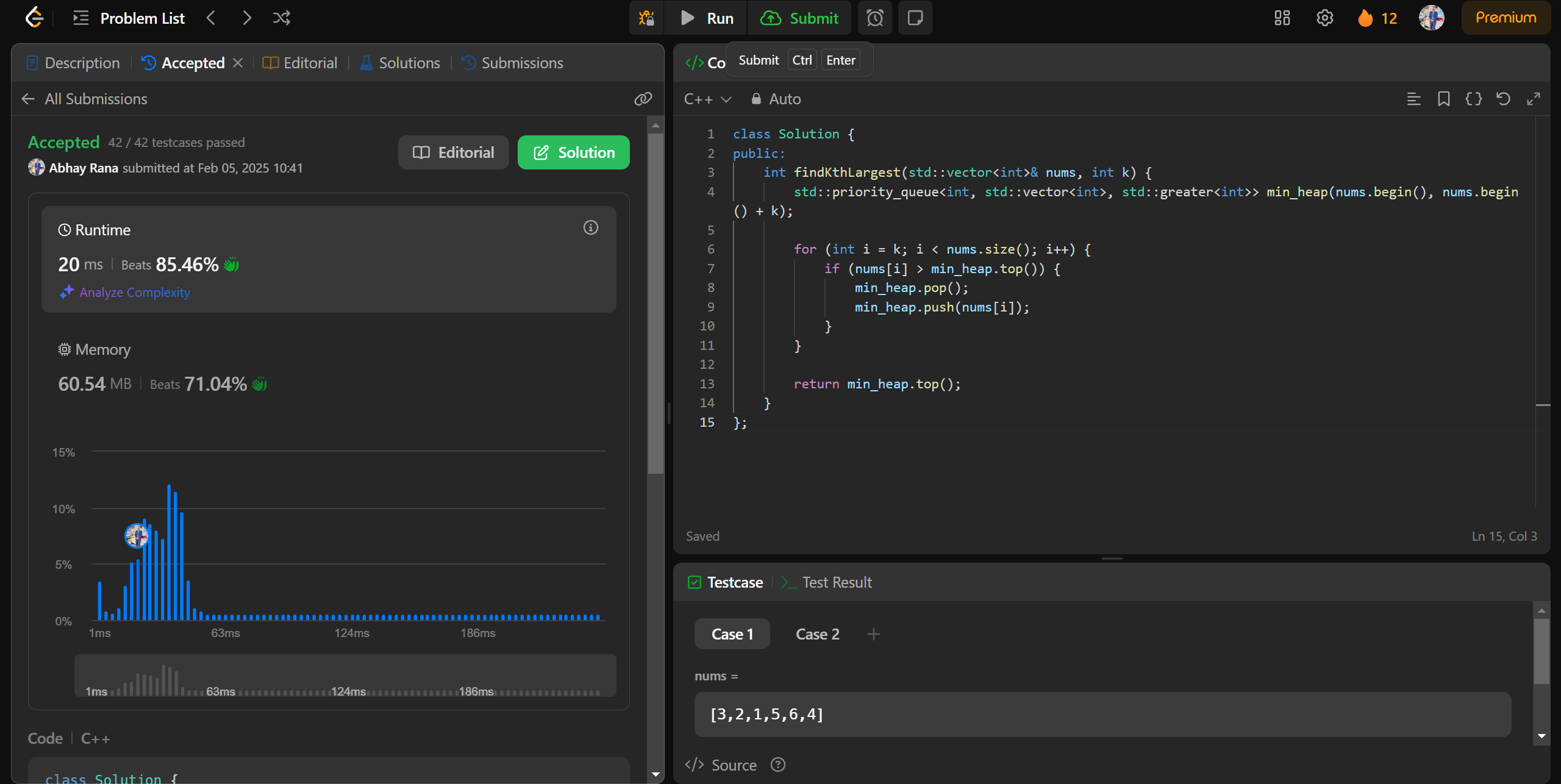
}

}

return min\_heap.top();

}

};



162.[Find Peak Element](https://leetcode.com/problems/find-peak-element/description/)

class Solution {

public:

int findPeakElement(vector<int>& nums) {

int left = 0;

int right = nums.size() - 1;

while (left < right) {

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

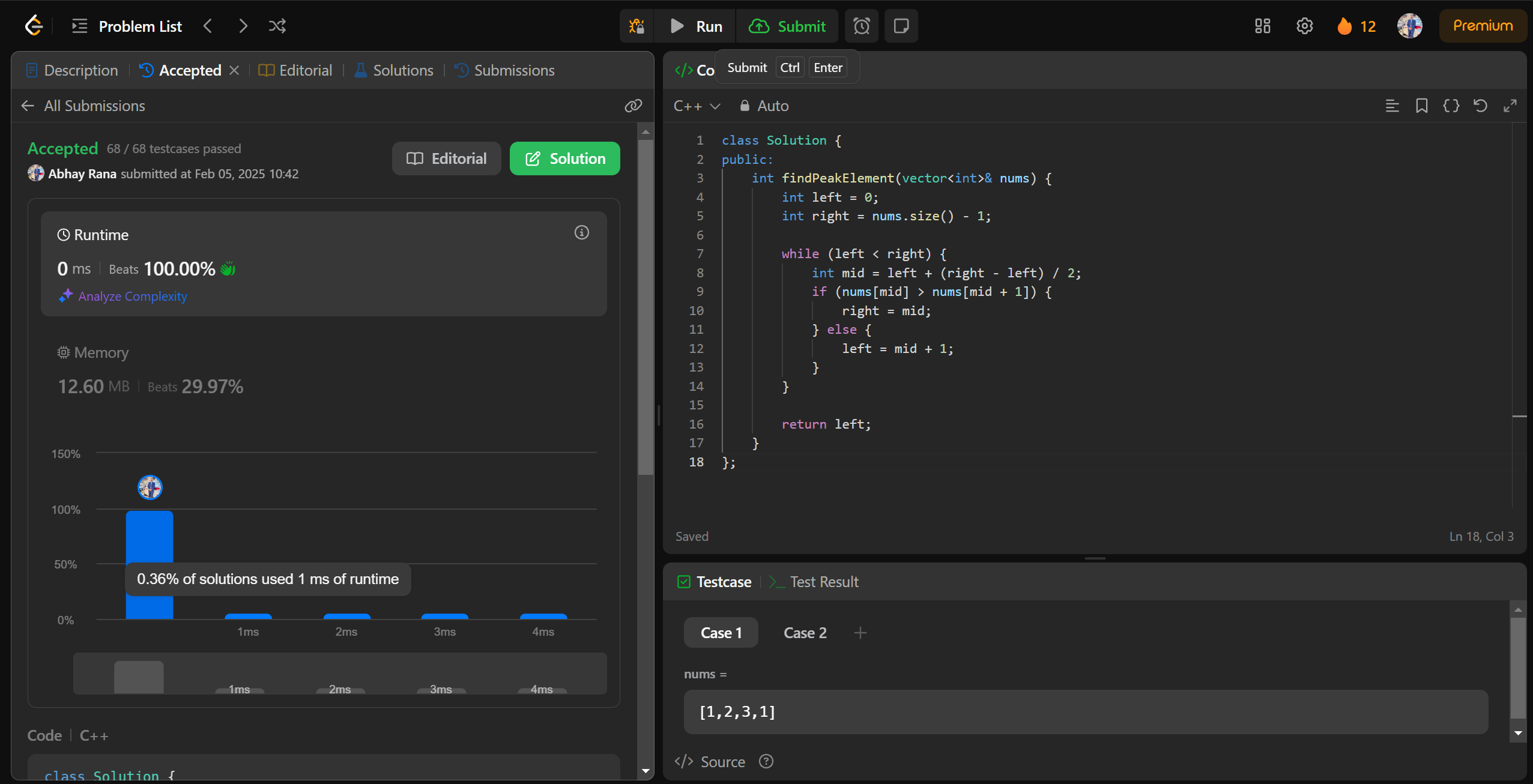
if (nums[mid] > nums[mid + 1]) {

right = mid;

} else {

left = mid + 1;

}}

return left; }};

56.[Merge Intervals](https://leetcode.com/problems/merge-intervals/description/)

class Solution {

public:

vector<vector<int>> merge(vector<vector<int>>& intervals) {

sort(intervals.begin(), intervals.end(), [](const vector<int>& a, const vector<int>& b) {

return a[0] < b[0];

});

vector<vector<int>> merged;

vector<int> prev = intervals[0];

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

vector<int> interval = intervals[i];

if (interval[0] <= prev[1]) {

prev[1] = max(prev[1], interval[1]);

} else {

merged.push\_back(prev);

prev = interval;

}

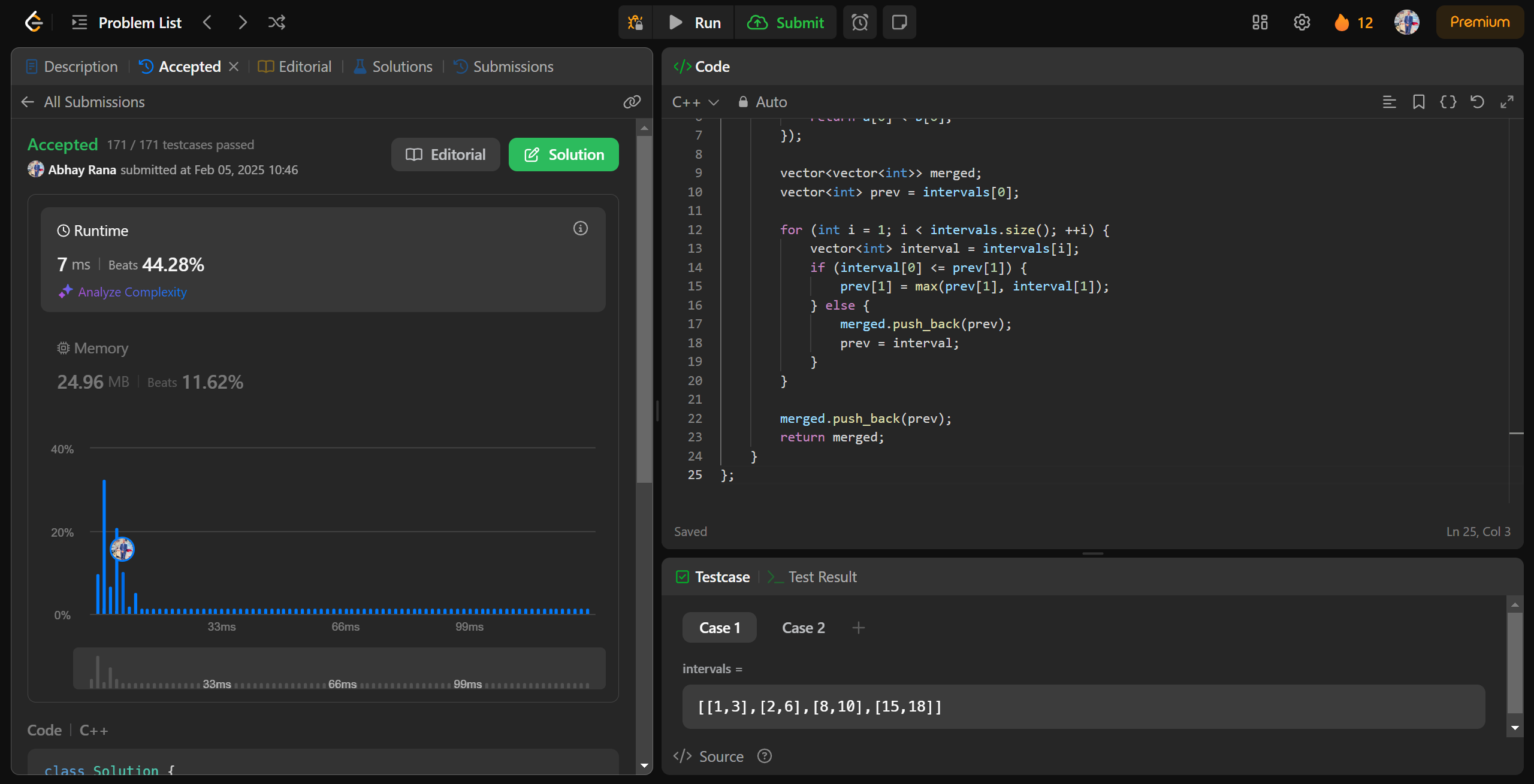
}

merged.push\_back(prev);

return merged;

}

};



33.[Search in Rotated Sorted Array](https://leetcode.com/problems/search-in-rotated-sorted-array/description/)

class Solution {

public:

int search(vector<int>& nums, int target) {

int left = 0;

int right = nums.size() - 1;

while (left <= right) {

int mid = (left + right) / 2;

if (nums[mid] == target) {

return mid;

} else if (nums[mid] >= nums[left]) {

if (nums[left] <= target && target <= nums[mid]) {

right = mid - 1;

} else {

left = mid + 1;

}

} else {

if (nums[mid] <= target && target <= nums[right]) {

left = mid + 1;

} else {

right = mid - 1;

}

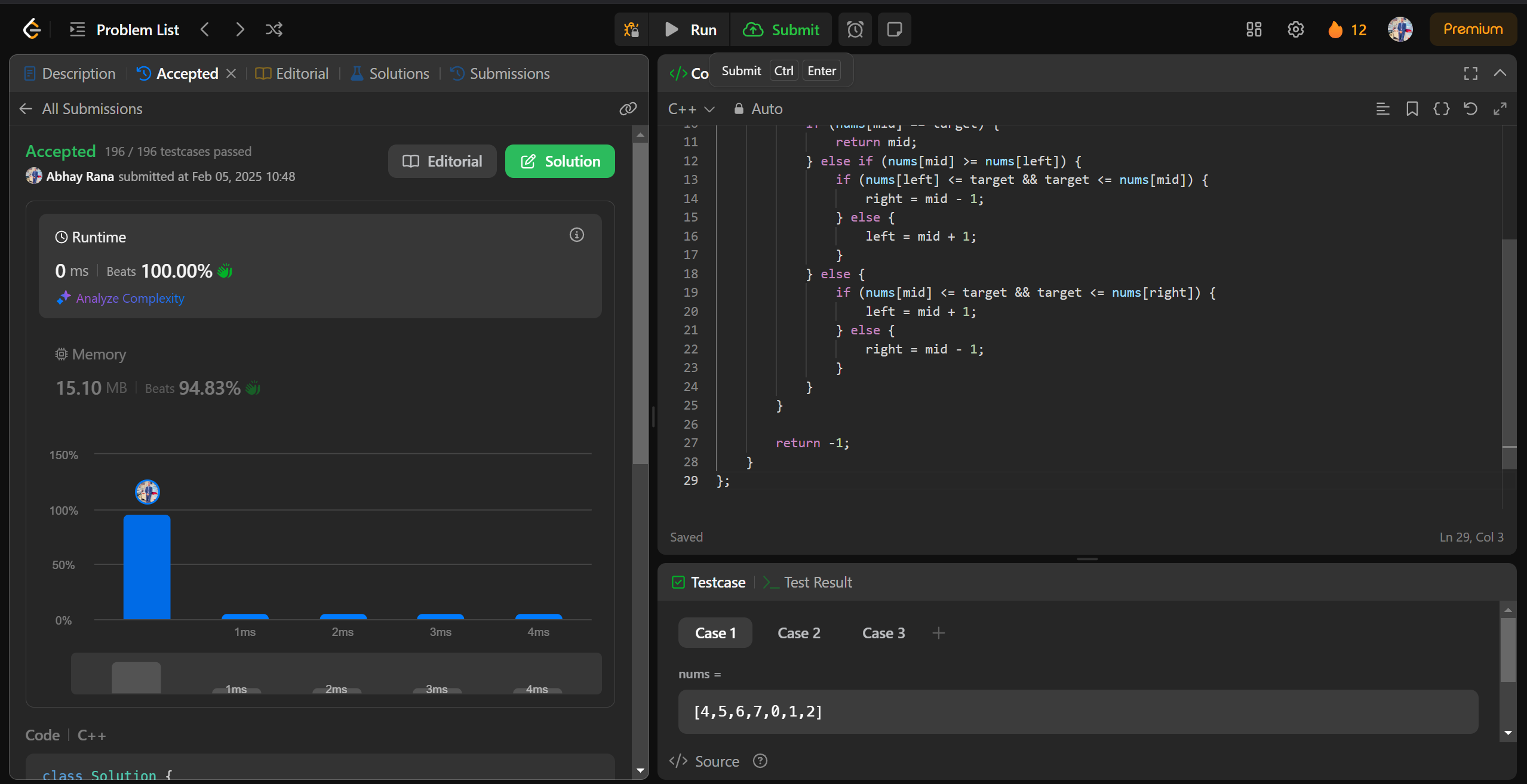
}

}

return -1;

}

};



240.[Search a 2D Matrix II](https://leetcode.com/problems/search-a-2d-matrix-ii/description/)

class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

int n = matrix.size(), m = matrix[0].size();

int row = 0, col = m - 1;

while (row < n && col >= 0) {

if (matrix[row][col] == target) return true;

else if (matrix[row][col] < target) row++;

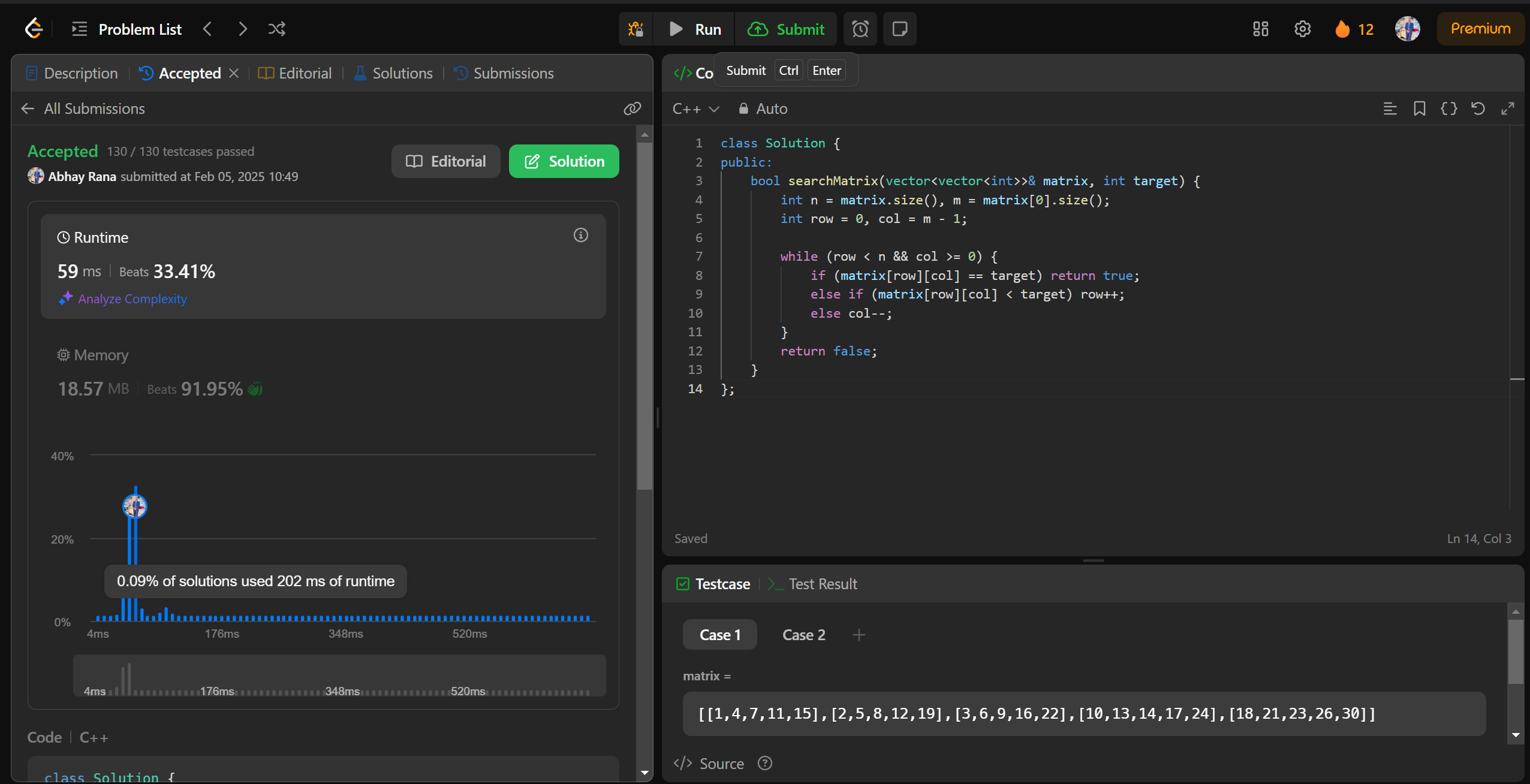
else col--;

}

return false;

}

};



324.[Wiggle Sort II](https://leetcode.com/problems/wiggle-sort-ii/description/)

class Solution {

public:

void wiggleSort(vector<int>& nums) {

int n = nums.size();

vector<int> nums1(nums);

sort(nums1.begin(), nums1.end());

int i = n-1;

int j = 0;

int k = i/2 + 1;

while(i >= 0)

{

if(i % 2 == 1)

{

nums[i] = nums1[k];

k++;

}

else

{

nums[i] = nums1[j];

j++;

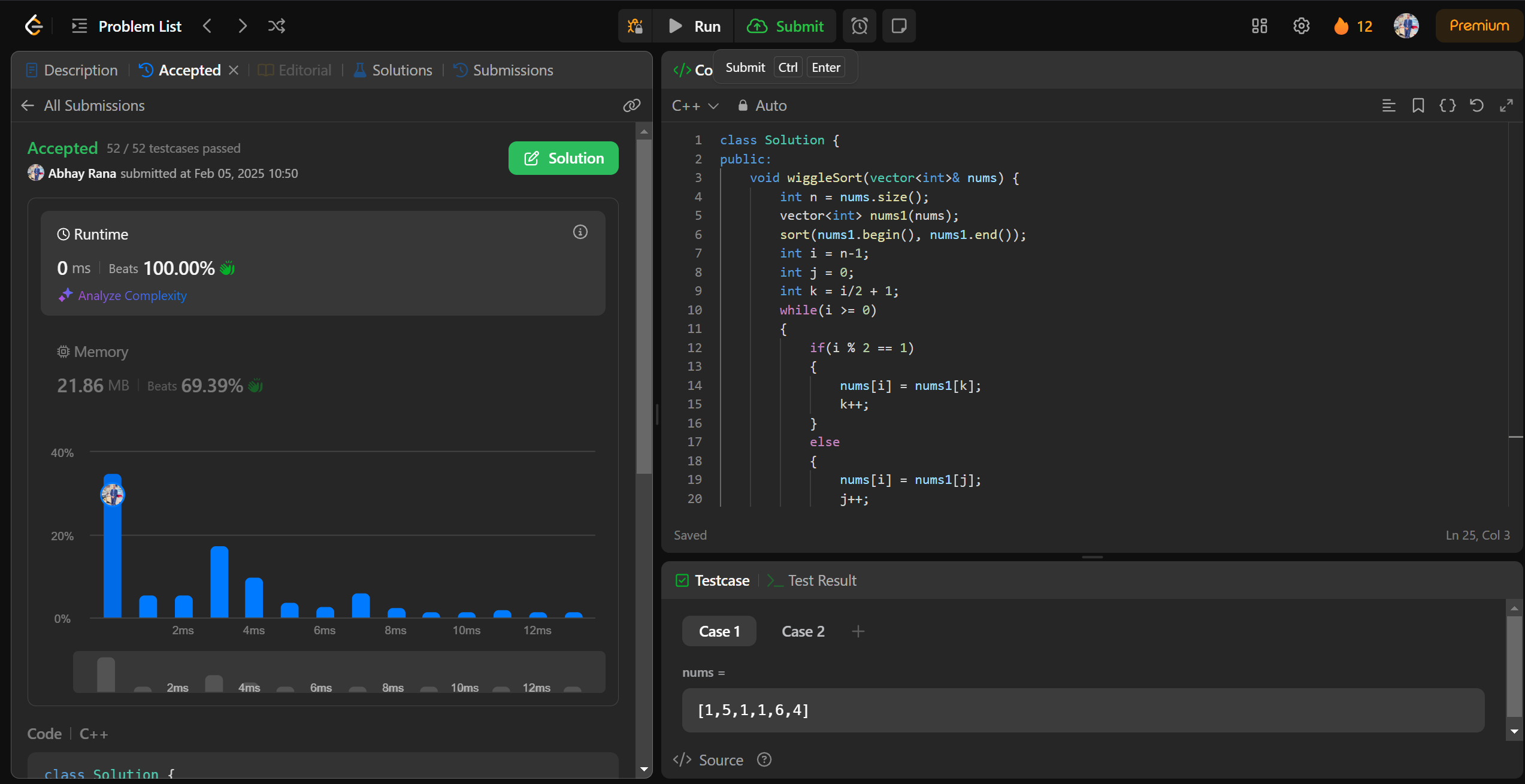
}

i--;

}

}

};



378.[Kth Smallest Element in a Sorted Matrix](https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/description/)

class Solution {

public:

int kthSmallest(vector<vector<int>>& matrix, int k) {

int n = matrix.size();

int m = matrix[0].size();

vector<int> p;

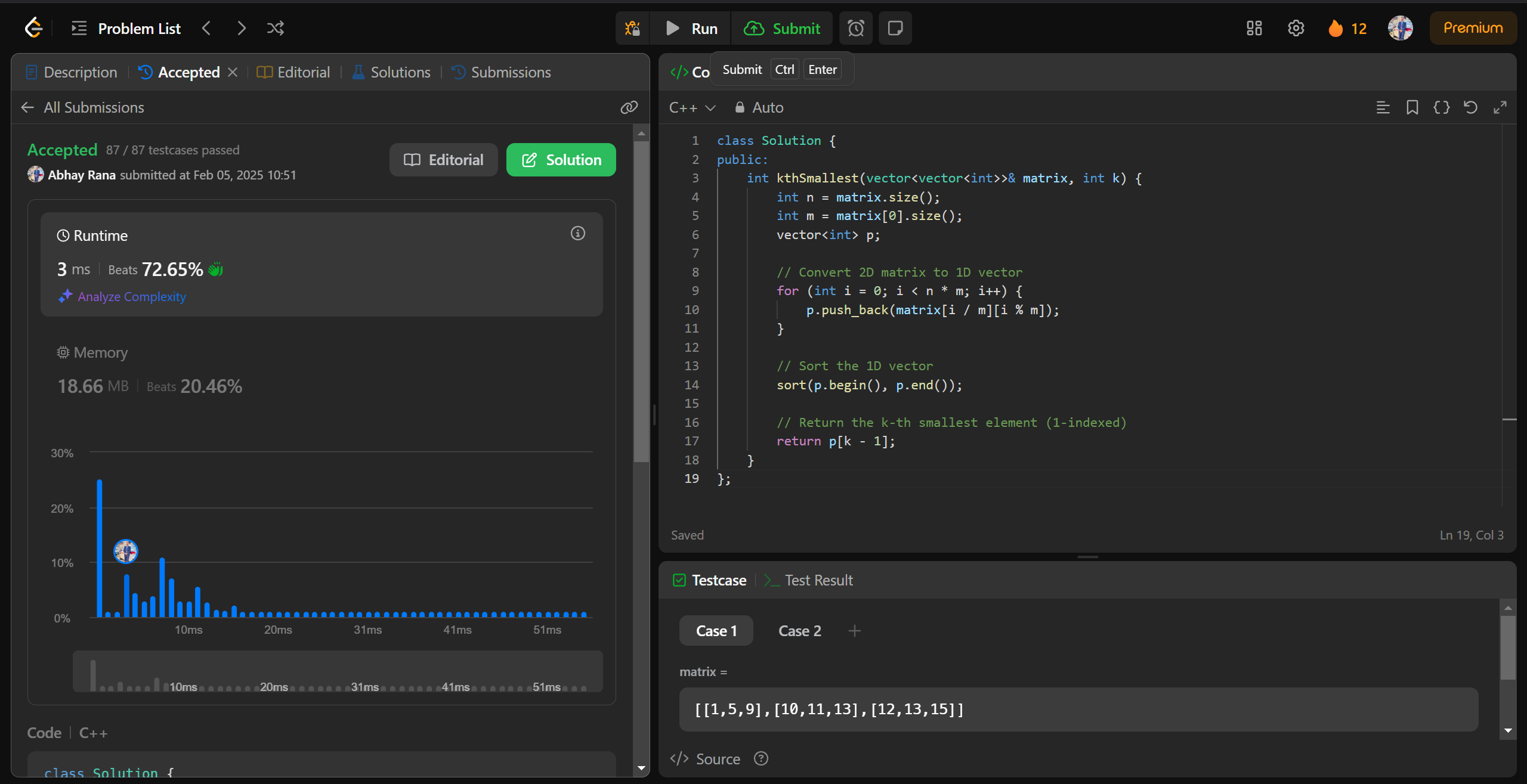
for (int i = 0; i < n \* m; i++) {

p.push\_back(matrix[i / m][i % m]);

}

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

return p[k - 1];

}};

4.[Median of Two Sorted Arrays](https://leetcode.com/problems/median-of-two-sorted-arrays/description/)

class Solution {

public:

    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {

        int n = nums1.size();

        int m = nums2.size();

        vector<int> merged;

        for (int i = 0; i < n; i++) {

            merged.push\_back(nums1[i]);

        }

        for (int i = 0; i < m; i++) {

            merged.push\_back(nums2[i]);

        }

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

        int total = merged.size();

        if (total % 2 == 1) {

            return static\_cast<double>(merged[total / 2]);

        } else {

            int middle1 = merged[total / 2 - 1];

            int middle2 = merged[total / 2];

            return (static\_cast<double>(middle1) + static\_cast<double>(middle2)) / 2.0;

        }

    }

};

