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

* **Solution Code:**

class Solution {

 public:

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

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

  }

 private:

  TreeNode\* build(const vector<int>& nums, int l, int r) {

    if (l > r)

      return nullptr;

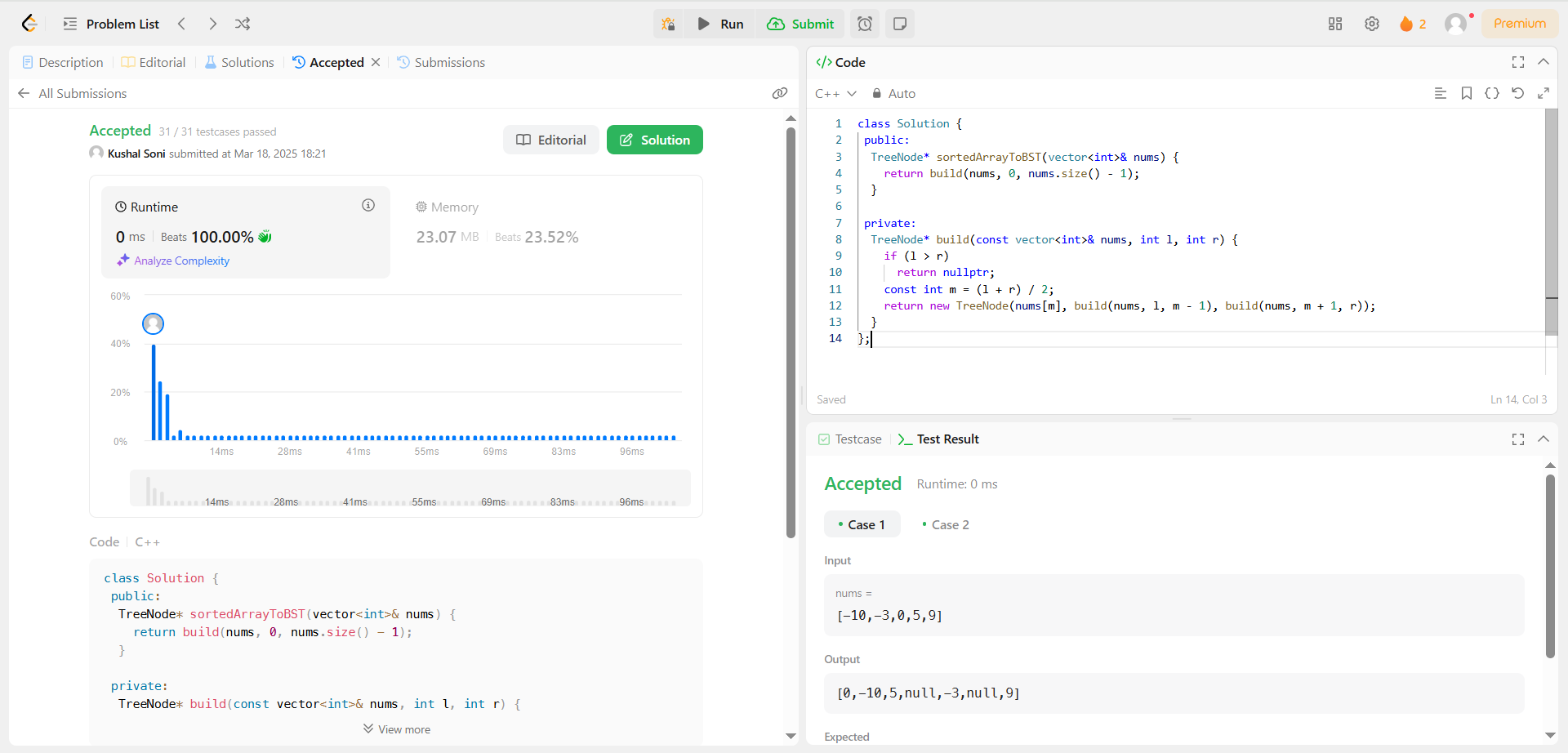
    const int m = (l + r) / 2;

    return new TreeNode(nums[m], build(nums, l, m - 1), build(nums, m + 1, r));

  }

};

* **Screenshot:**

****

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

* **Solution Code:**

class Solution {

public:

    int hammingWeight(int n) {

        int c=0;

        while(n){

            int r=n&1;

            n>>=1;

            if(r){

                c++;

            }

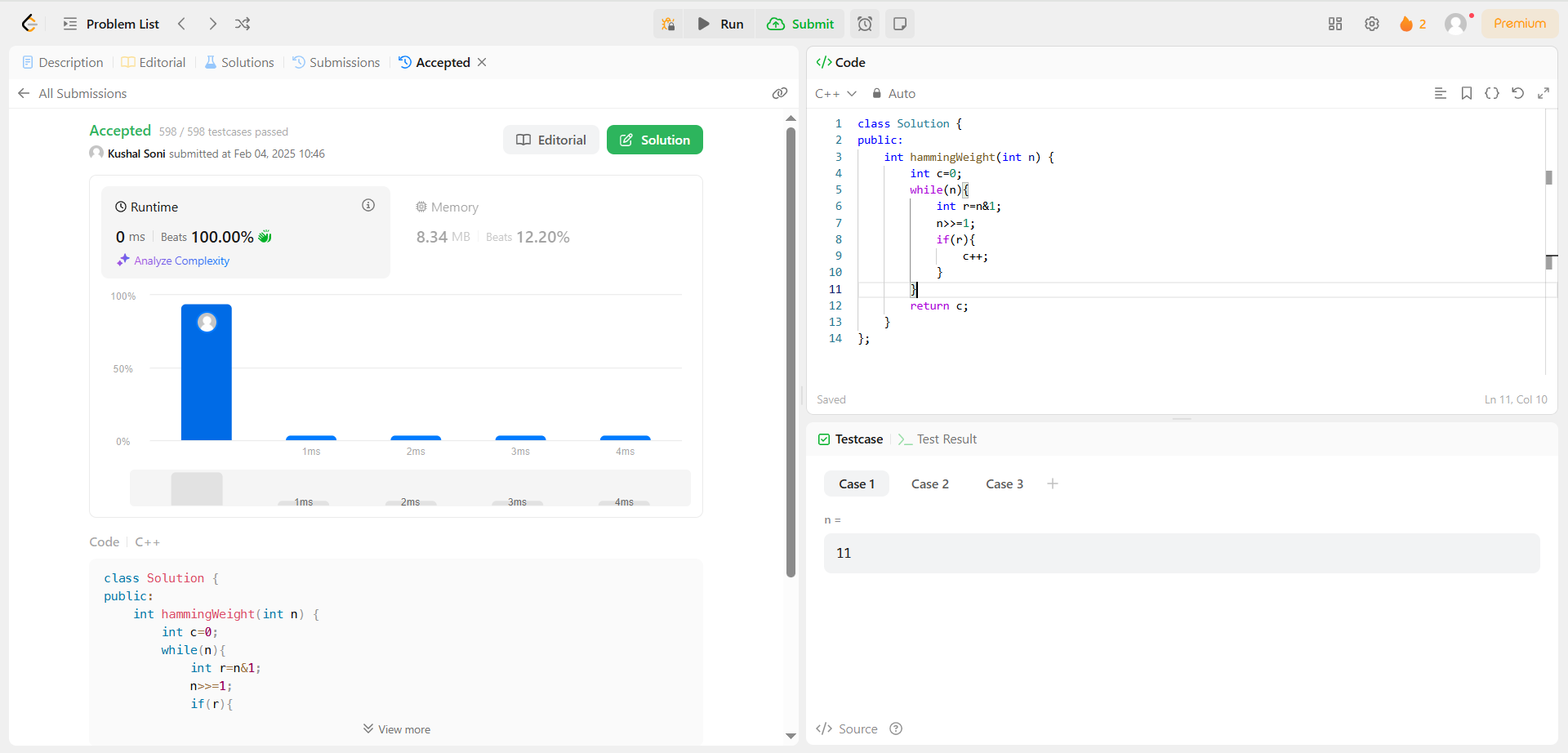
        }

        return c;

    }

};

* **Screenshot:**

****

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

* **Solution Code:**

class Solution {

public:

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

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

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

        vector<int> counter(max-min+1);

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

            ++counter[nums[i]-min];

        }

        int j = 0;

        for (int x = min; x <= max; ++x){

            for (int i = 1; i <= counter[x - min]; ++ i){

                nums[j++] = x;

            }

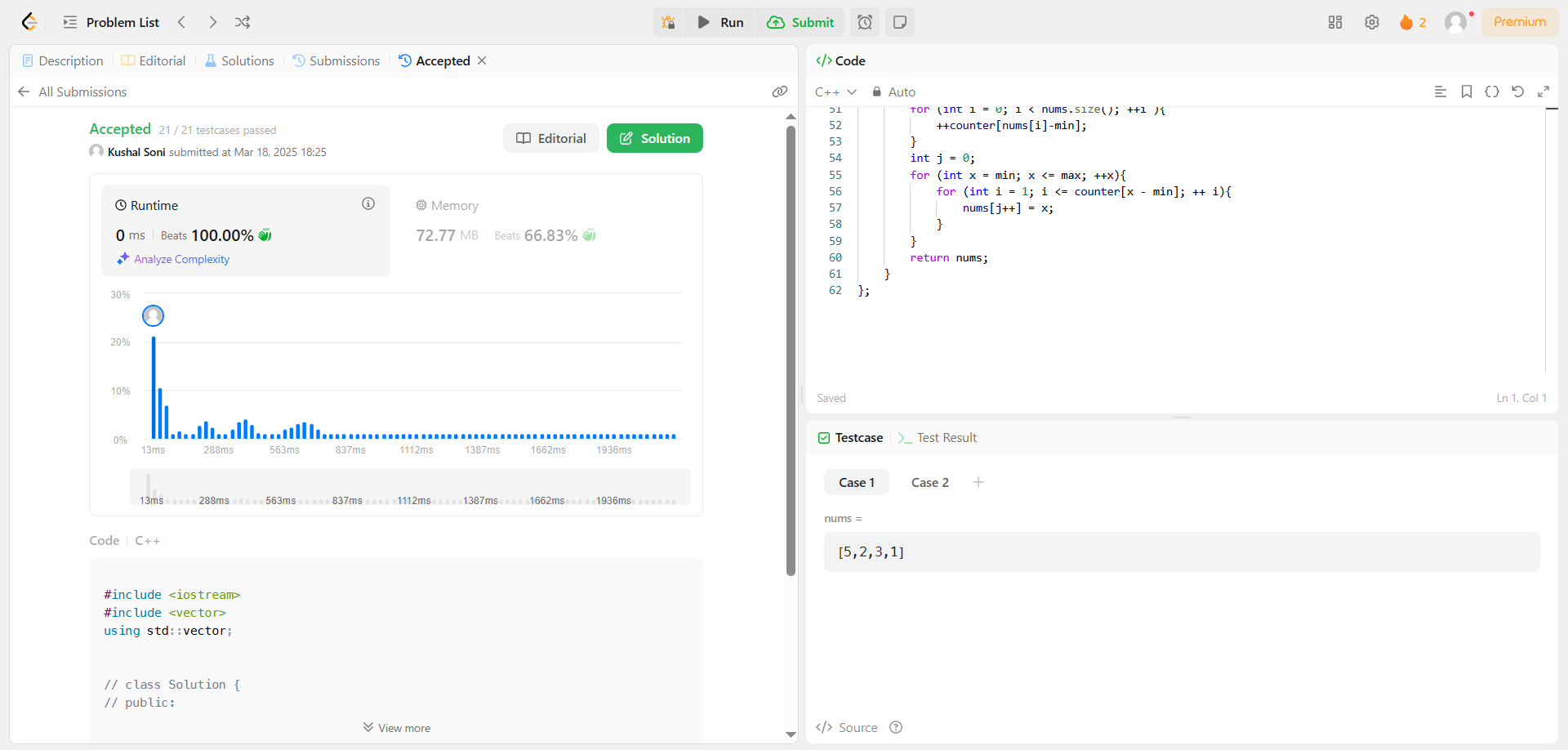
        }

        return nums;

    }

};

* **Screenshot:**

****

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

* **Solution Code:**

class Solution {

public:

    int maxSubArray(vector<int>& nums) {

        int res = nums[0];

        int total = 0;

        for (int n : nums) {

            if (total < 0)

                total = 0;

            total += n;

            res = max(res, total);

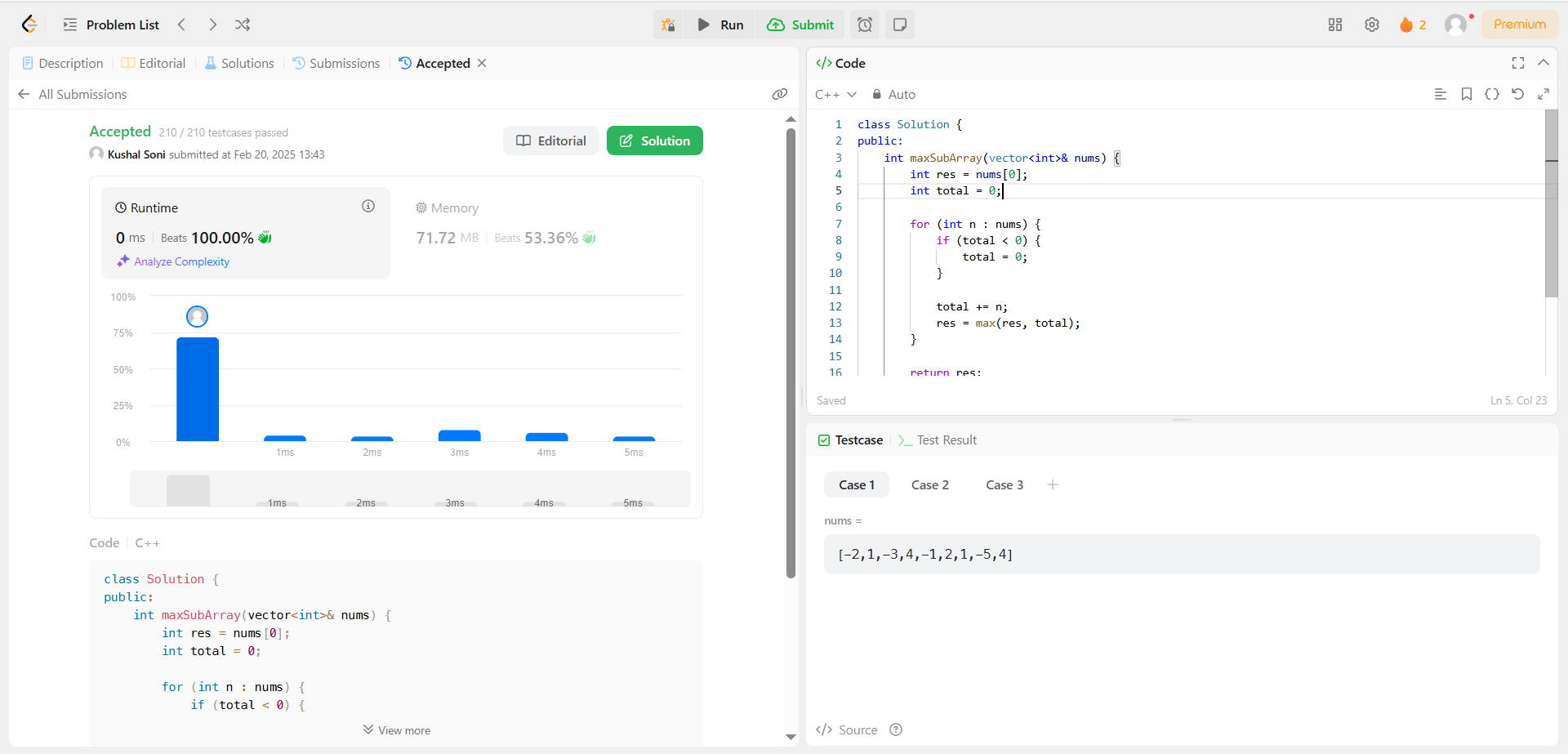
        }

        return res;

    }

};

* **Screenshot:**

****

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

* **Solution Code:**

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);

}

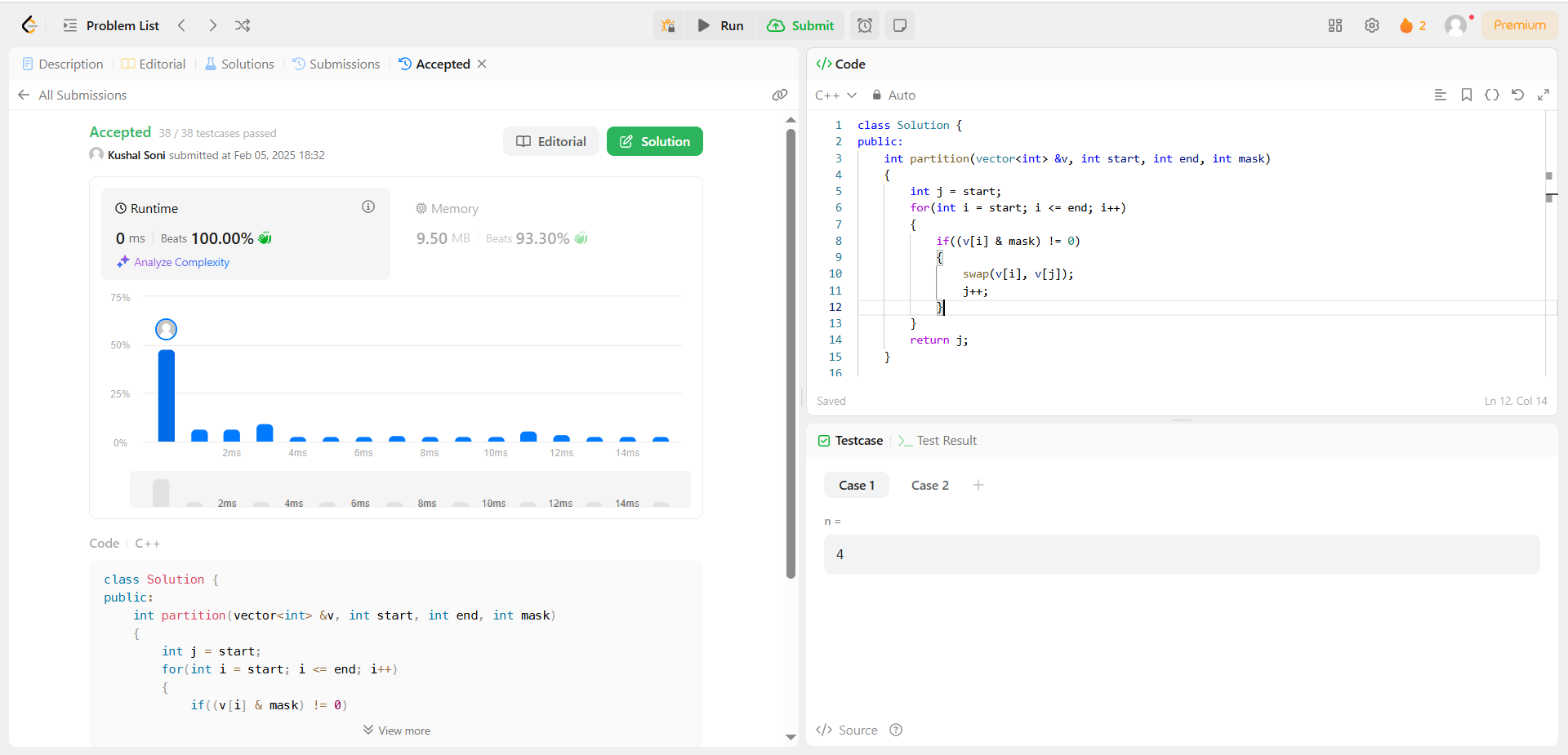
        sort(ans, 0, N - 1, 1);

        return ans;

    }

};

* **Screenshot:**

****

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

* **Solution Code:**

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)

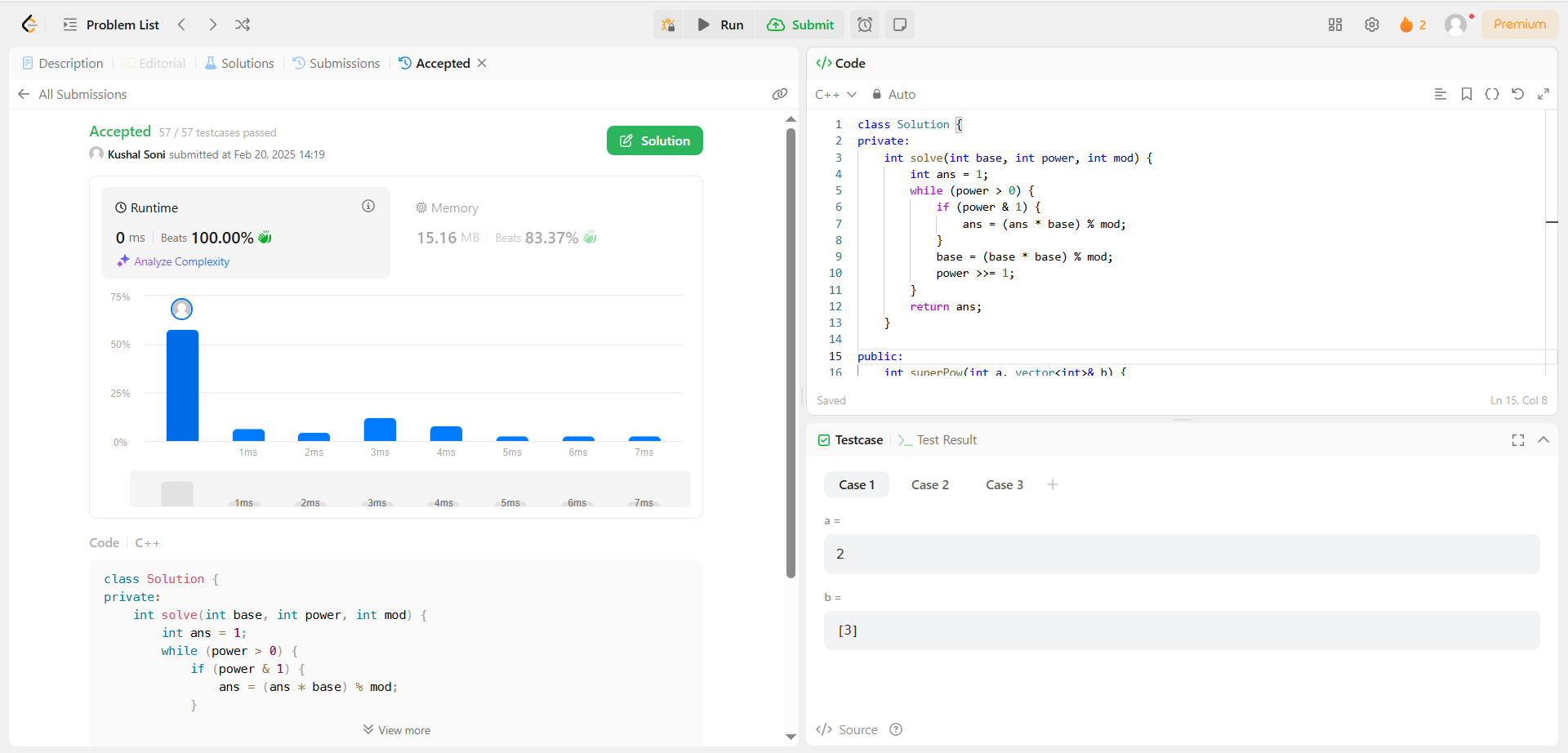
            expi = m;

        return solve(a,expi,1337);

    }

};

* **Screenshot:**

****

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

* **Solution Code:**

const auto \_ = std::cin.tie(nullptr)->sync\_with\_stdio(false);

#define LC\_HACK

#ifdef LC\_HACK

const auto \_\_ = []() {

    struct \_\_\_ {

        static void \_() { std::ofstream("display\_runtime.txt") << 0 << '\n'; }

    };

    std::atexit(&\_\_\_::\_);

    return 0;

}();

#endif

struct Corner {

    int x;

    int y;

    bool operator < (const Corner& other) const {

        if (x == other.x)

            return y < other.y;

        return x < other.x;

    }

};

class Solution {

public:

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

        vector<Corner> all;

        vector<vector<int>> result;

        all.reserve(buildings.size() \* 2);

        for (const auto& building : buildings) {

            all.emplace\_back(building[0], -building[2]);

            all.emplace\_back(building[1], building[2]);

        }

        sort(all.begin(), all.end());

        priority\_queue<int> heights;

        unordered\_map<int, int> removed;

        int maxHeight = 0;

        heights.push(0);

        for (const auto& vertice : all) {

            if (vertice.y < 0)

                heights.push(-vertice.y);

            else

                removed[vertice.y]++;

            while (!heights.empty() && removed[heights.top()] > 0) {

                removed[heights.top()]--;

                heights.pop();

            }

            if (maxHeight != heights.top()) {

                maxHeight = heights.top();

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

            }

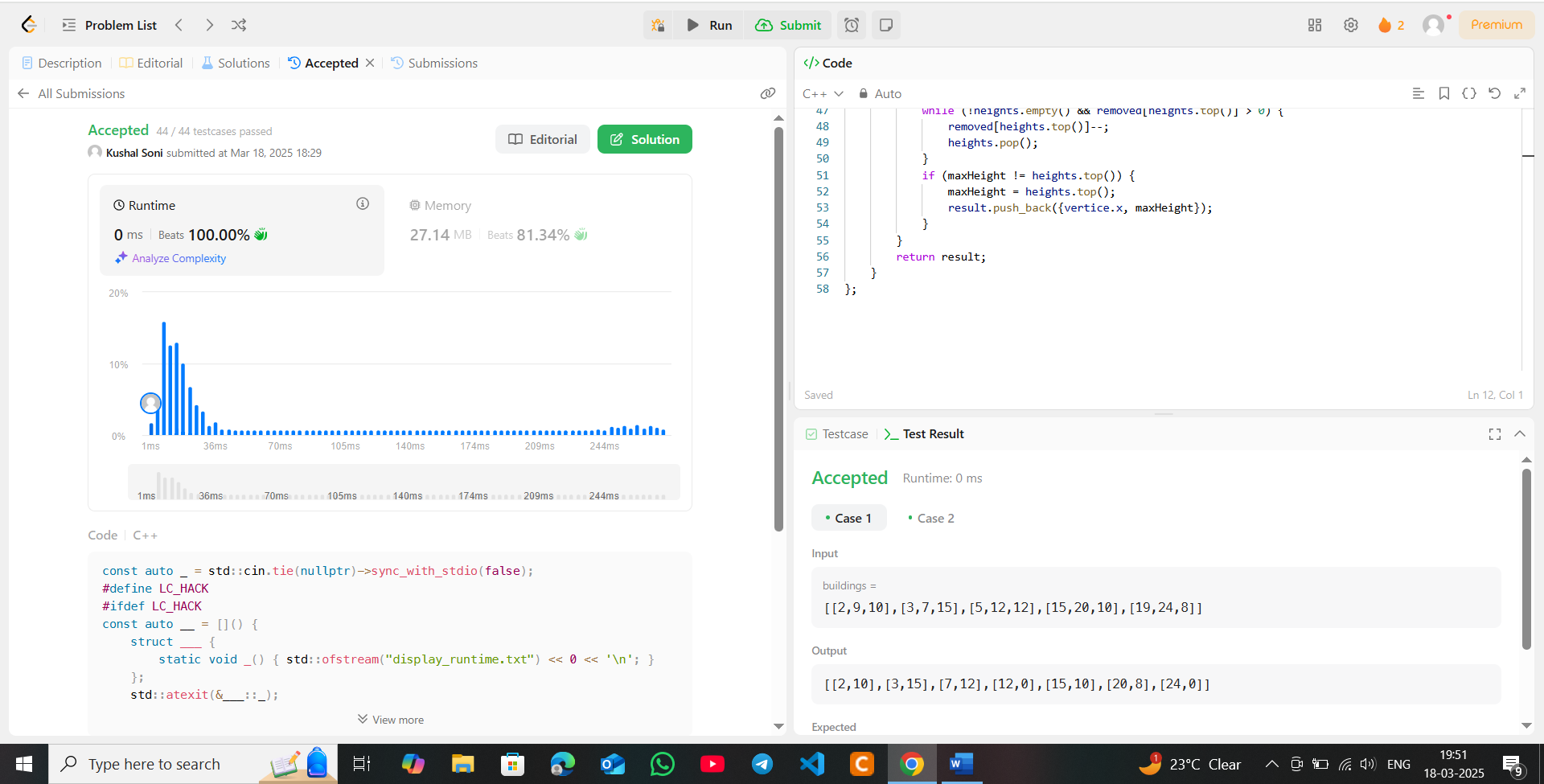
        }

        return result;

    }

};

* **Screenshot:**

****