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

* **Solution Code:**

class Solution {

public:

    string longestNiceSubstring(string s) {

        string output = "";

        int count = 0;

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

            int smallMask=0;

            int largeMask = 0;

            char ch = s[i];

            int chint = 0;

            if(ch>=65 && ch<=90){

                chint = ch-'A';

                largeMask = 1<<chint;

            }

            else{

                chint = ch-'a';

                smallMask = 1<<chint;

            }

            for(int j = i+1;j<s.length();j++){

                ch = s[j];

                if(ch>=65 && ch<=90){

                    chint = ch-'A';

                    largeMask |= 1<<chint;

                }

                else{

                    chint = ch-'a';

                    smallMask |= 1<<chint;

                }

                //checking for nice

                if((smallMask^largeMask) == 0){

                    if(count<j-i+1){

                        count = j-i+1;

                        string temp(s.begin()+i,s.begin()+j+1);

                        output = temp;

                    }

                }

            }

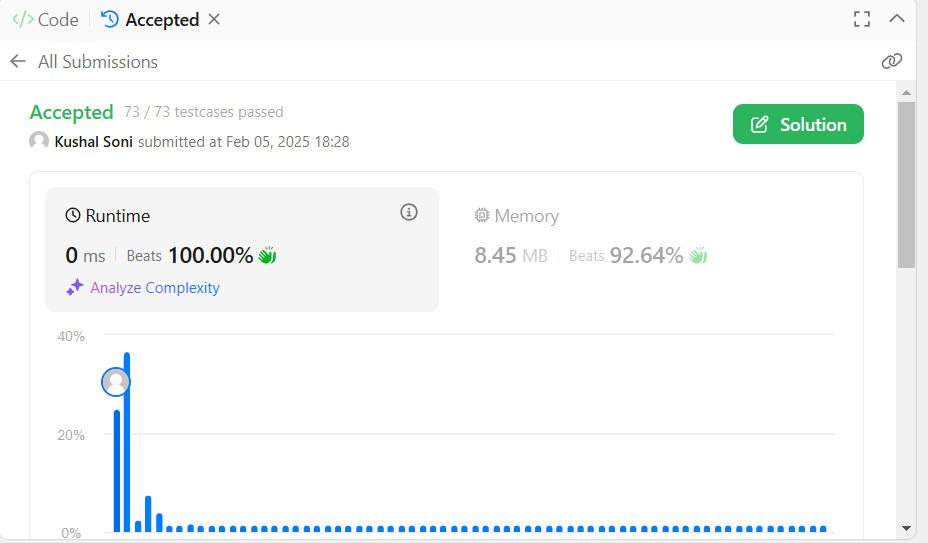
        }

        return output;

    }

};

* **Submission:**

****

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

* **Solution Code:**

class Solution {

public:

    uint32\_t reverseBits(uint32\_t n) {

        uint32\_t result = 0;

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

            int bit = n & 1;

            result = (result << 1) | bit;

            n = n >> 1;

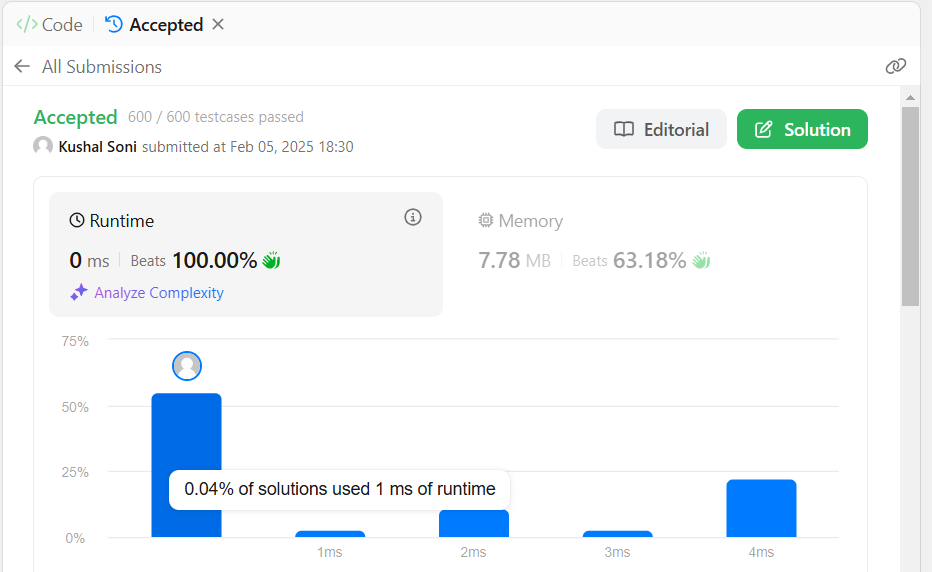
        }

        return result;

    }

};

* **Submission:**

****

**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++;

            }

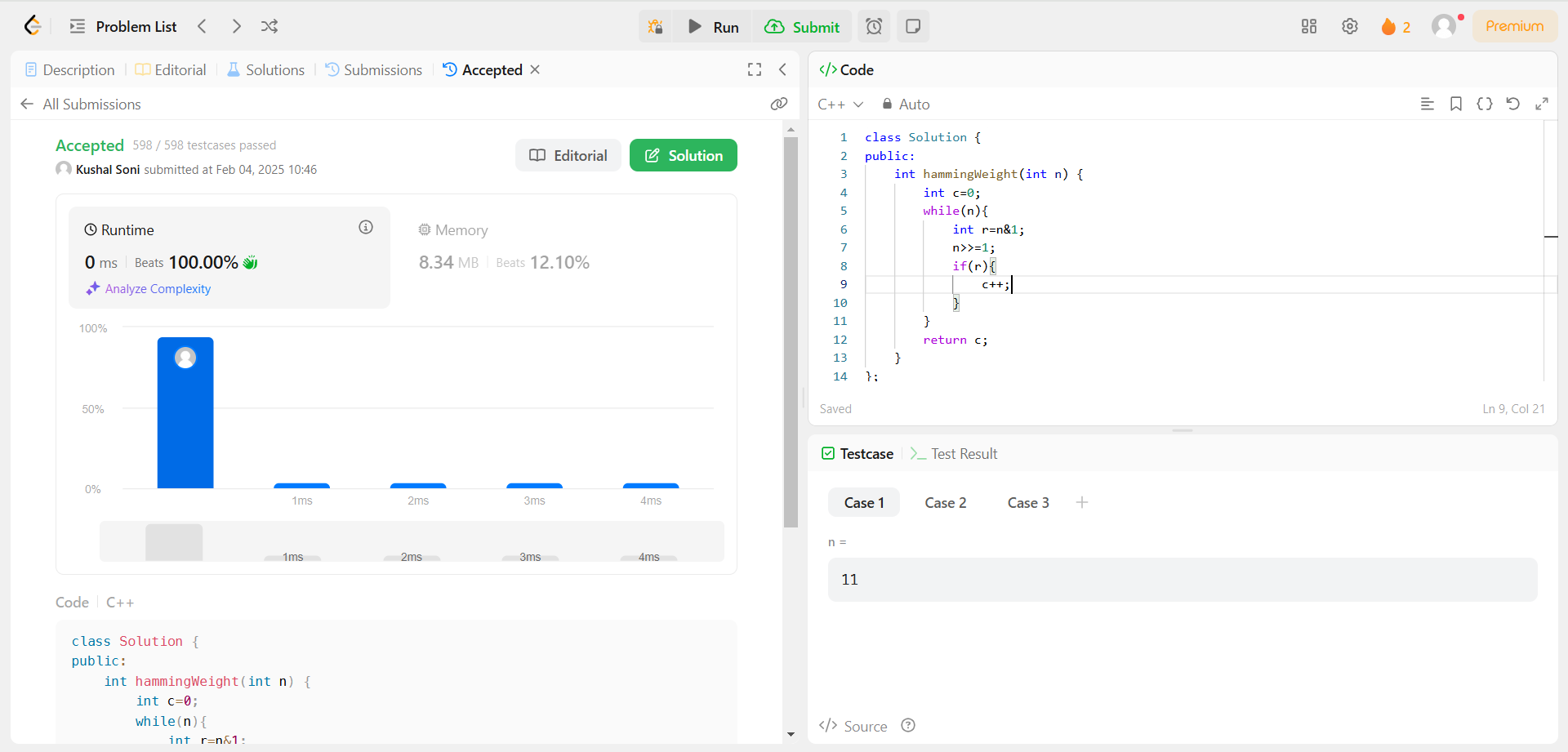
        }

        return c;

    }

};

* **Submission:**



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

#define pb push\_back

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--;

            }

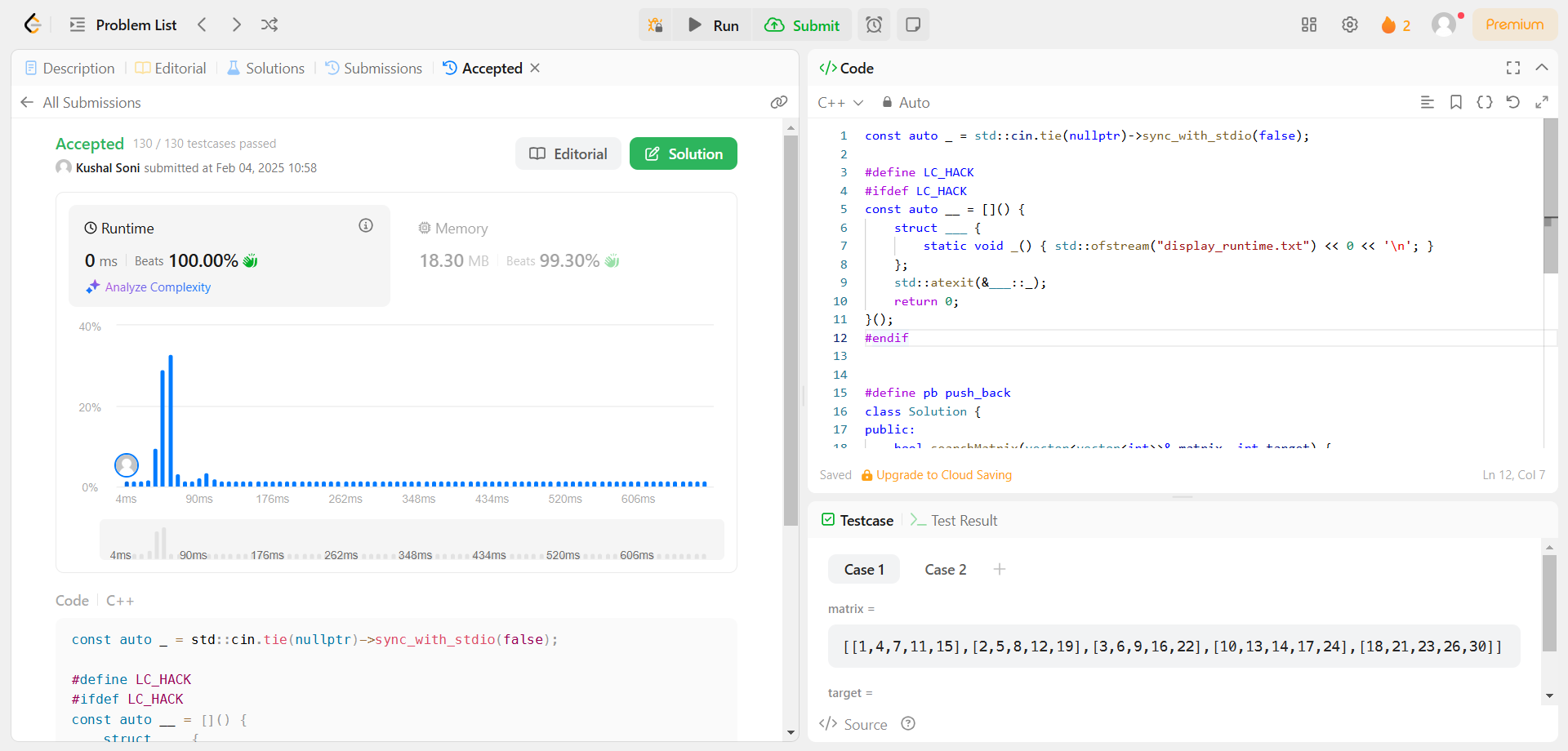
        }

        return false;

    }

};

* **Submission:**

****

**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);

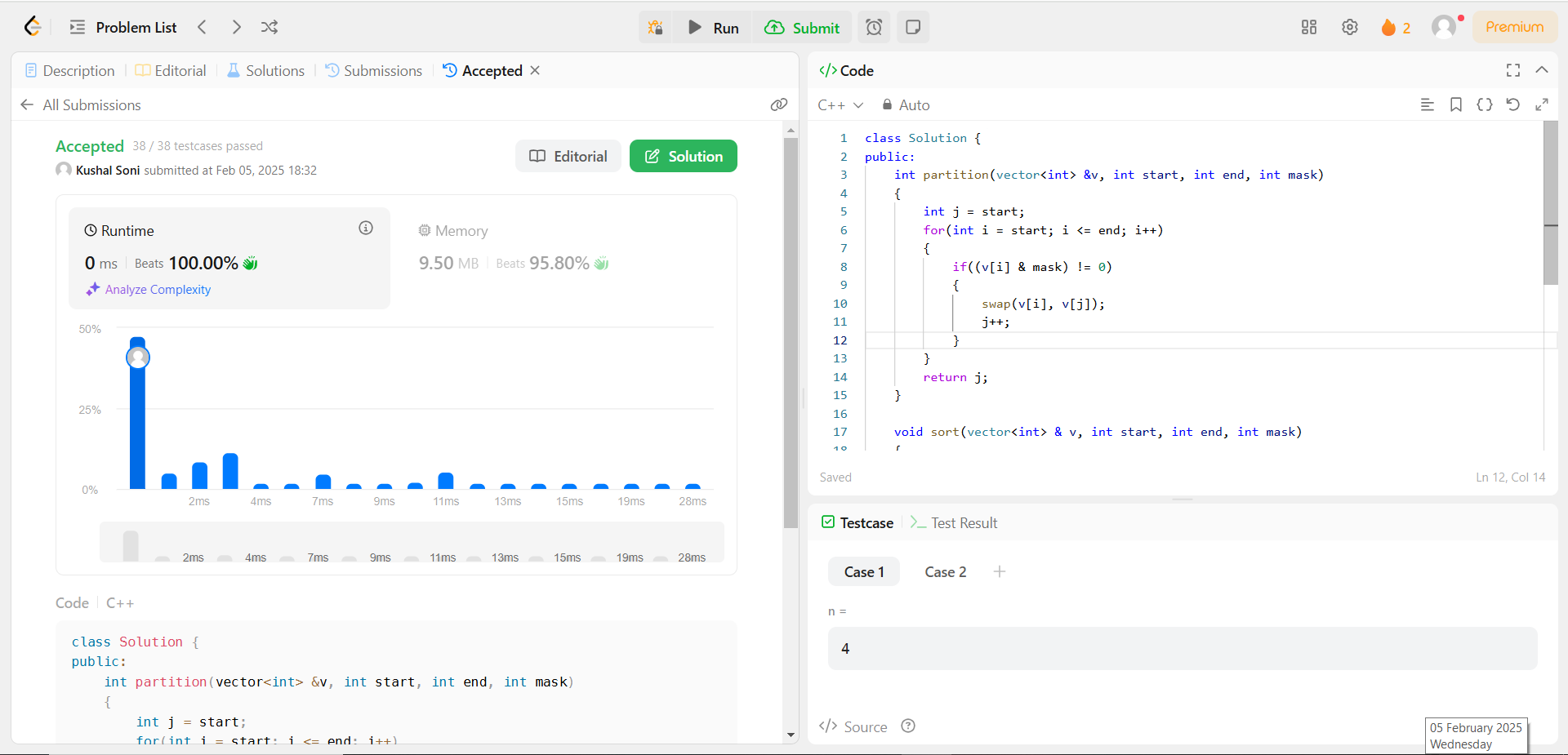
        sort(ans, 0, N - 1, 1);

        return ans;

    }

};

* **Submission:**

****

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

* **Solution Code:**

class Solution {

public:

    vector<int>tree;

    void update(int node,int st,int end,int i,int val){

        if(st==end){

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

            return;

        }

        int mid=(st+end)/2;

        if(i<=mid){

            update(node\*2,st,mid,i,val);

        }else{

            update(node\*2+1,mid+1,end,i,val);

        }

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

    }

    int query(int node,int st,int end,int x,int y){

        if(x>end || y<st) return -1e9;

        if(st>=x && end<=y){

            return tree[node];

        }

        int mid=(st+end)/2;

        int left=query(2\*node,st,mid,x,y);

        int right=query(2\*node+1,mid+1,end,x,y);

        return max(left,right);

    }

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

        int n=nums.size();

        if(n==1) return 1;

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

        tree.clear();

        tree.resize(4\*m+10);

        for(int i=n-1;i>=0;i--){

            int l=nums[i]+1,r=min(nums[i]+k,m);

            int x=query(1,0,m,l,r);

            if(x==-1e9) x=0;

            update(1,0,m,nums[i],x+1);

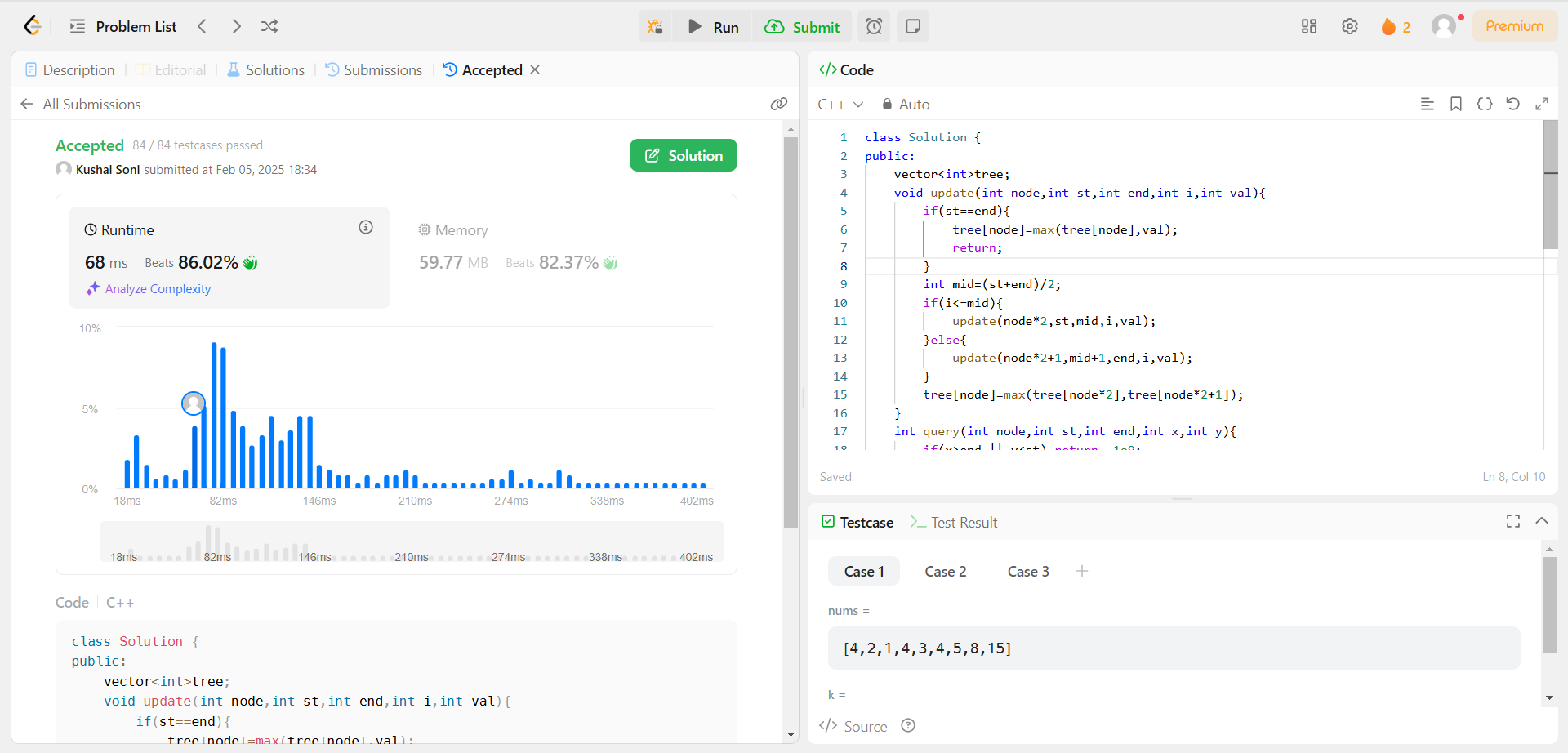
        }

        return tree[1];

    }

};

* **Submission:**

****

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

* **Solution Code:**

class Solution {

public:

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

        for(int i=m; i<m+n; i++){

            nums1[i]=nums2[i-m];

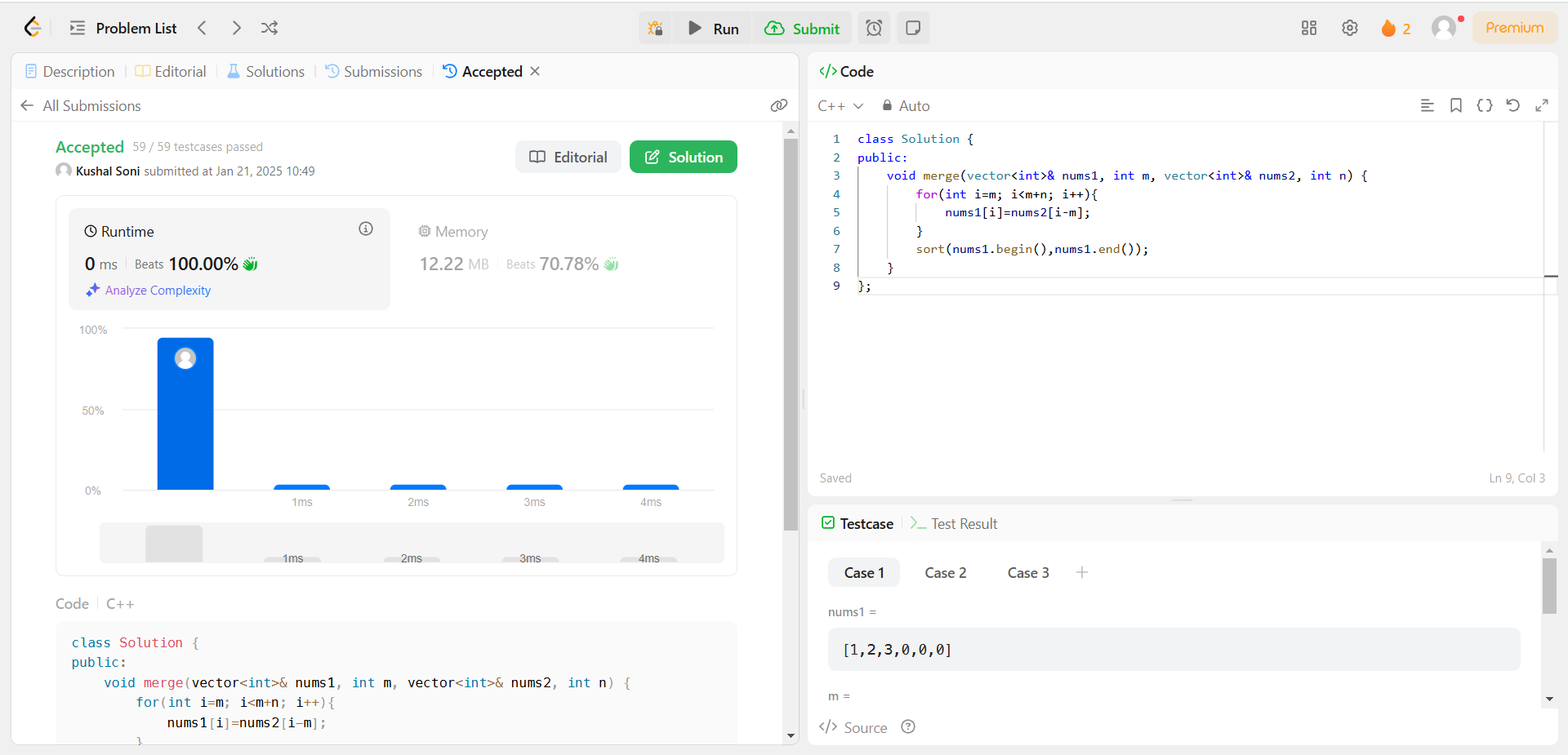
        }

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

    }

};

* **Submission:**

****

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

* **Solution Code:**

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;

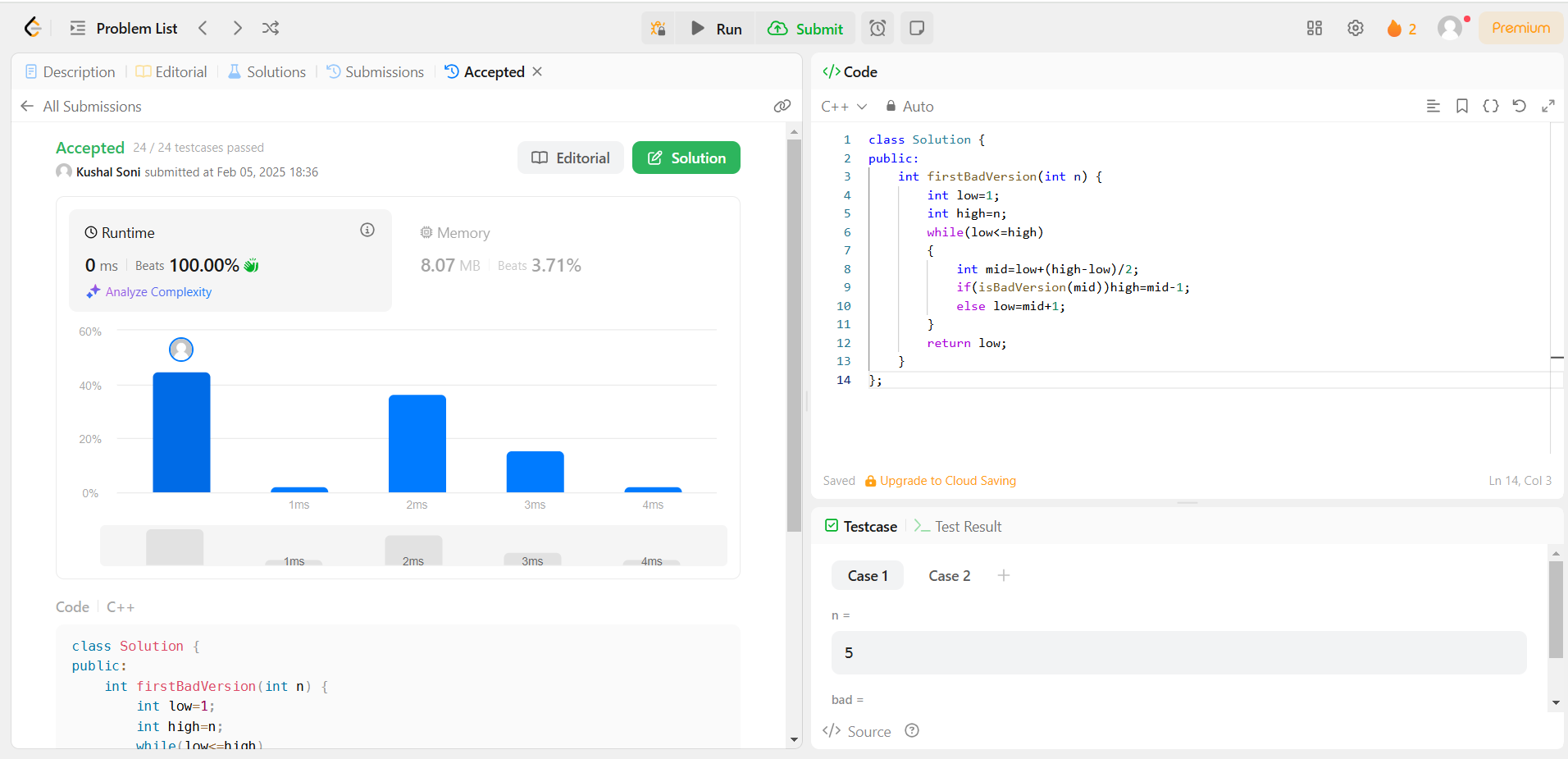
        }

        return low;

    }

};

* **Submission:**

****

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

* **Solution Code:**

class Solution {

public:

    void sortColors(vector<int>& nums) {

        int i=0;

        int j=0;

        int k=nums.size()-1;

        while(j<=k){

            if(nums[j]==0){

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

                i++;

                j++;

            }

            else if(nums[j]==1){

                j++;

            }

            else{

                swap(nums[j],nums[k]);

                k--;

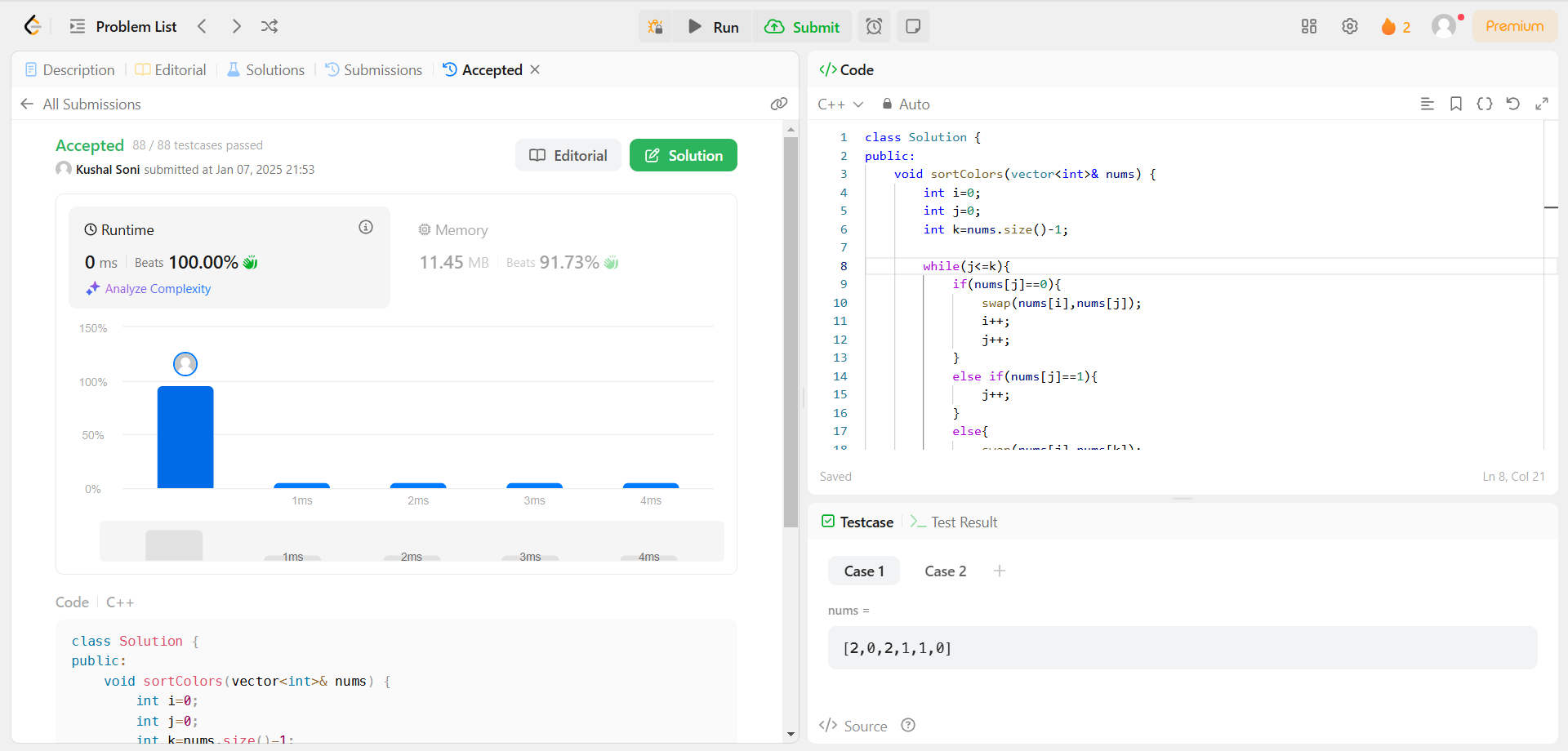
            }

        }

    }

};

* **Submission:**



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

* **Solution Code:**

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();

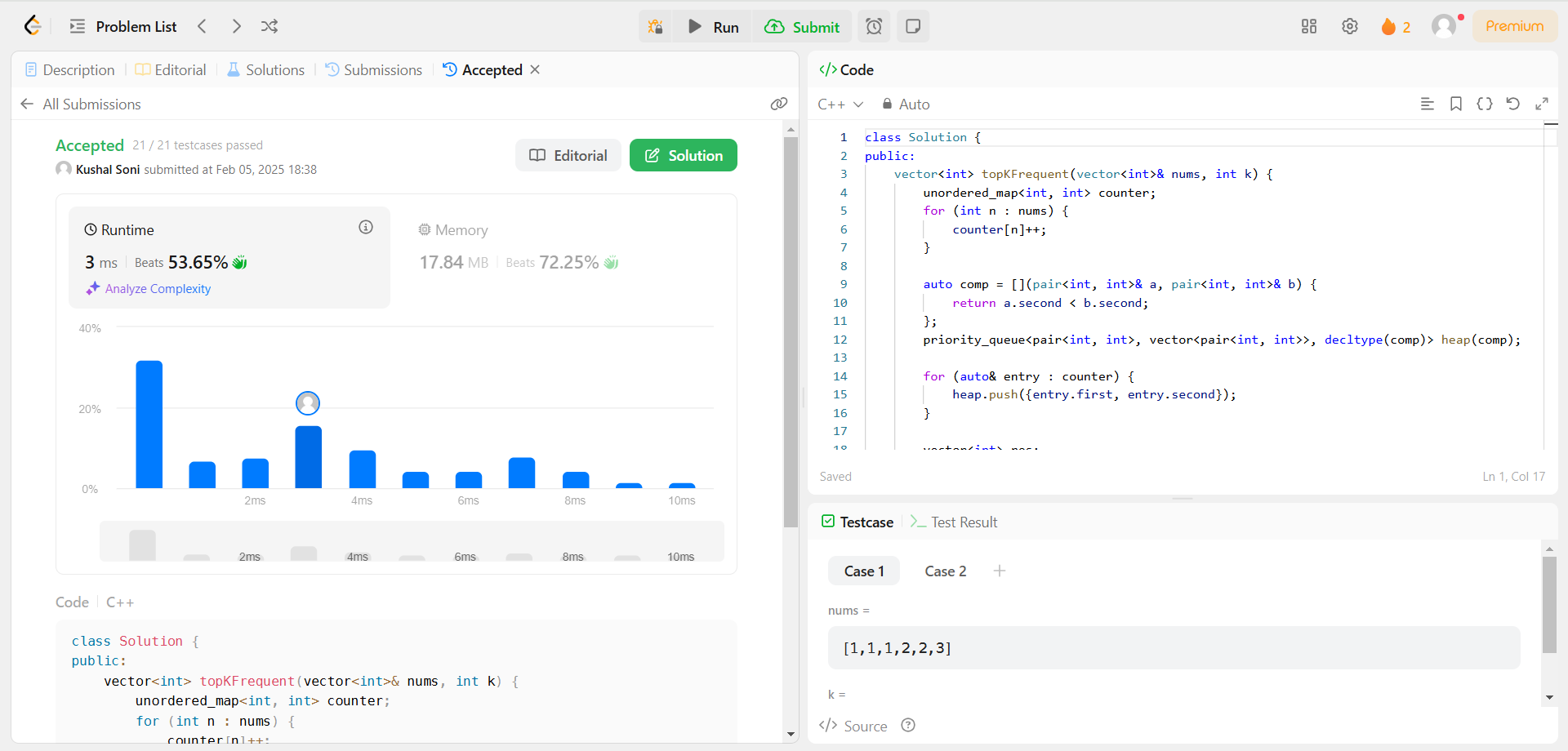
        }

        return res;

    }

};

* **Submission:**



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

* **Solution Code:**

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;

            }

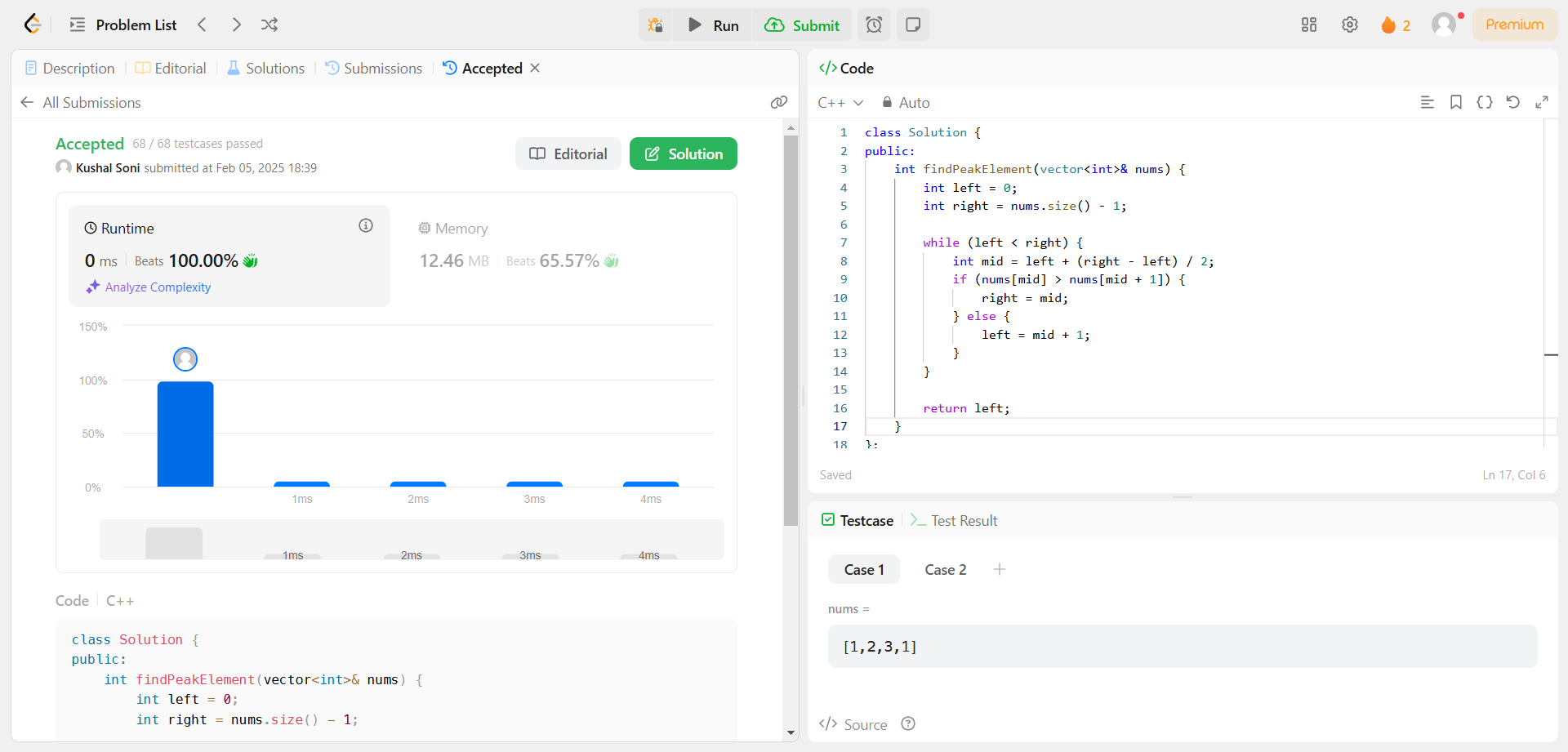
        }

        return left;

    }

};

* **Submission:**

****

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

* **Solution Code:**

class Solution {

public:

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

        vector<vector<int>> ans;

        ranges::sort(intervals);

        for(const vector<int>&interval:intervals){

            if(ans.empty()||ans.back()[1]<interval[0]){

                ans.push\_back(interval);

            }

            else

            ans.back()[1] = max(ans.back()[1], interval[1]);

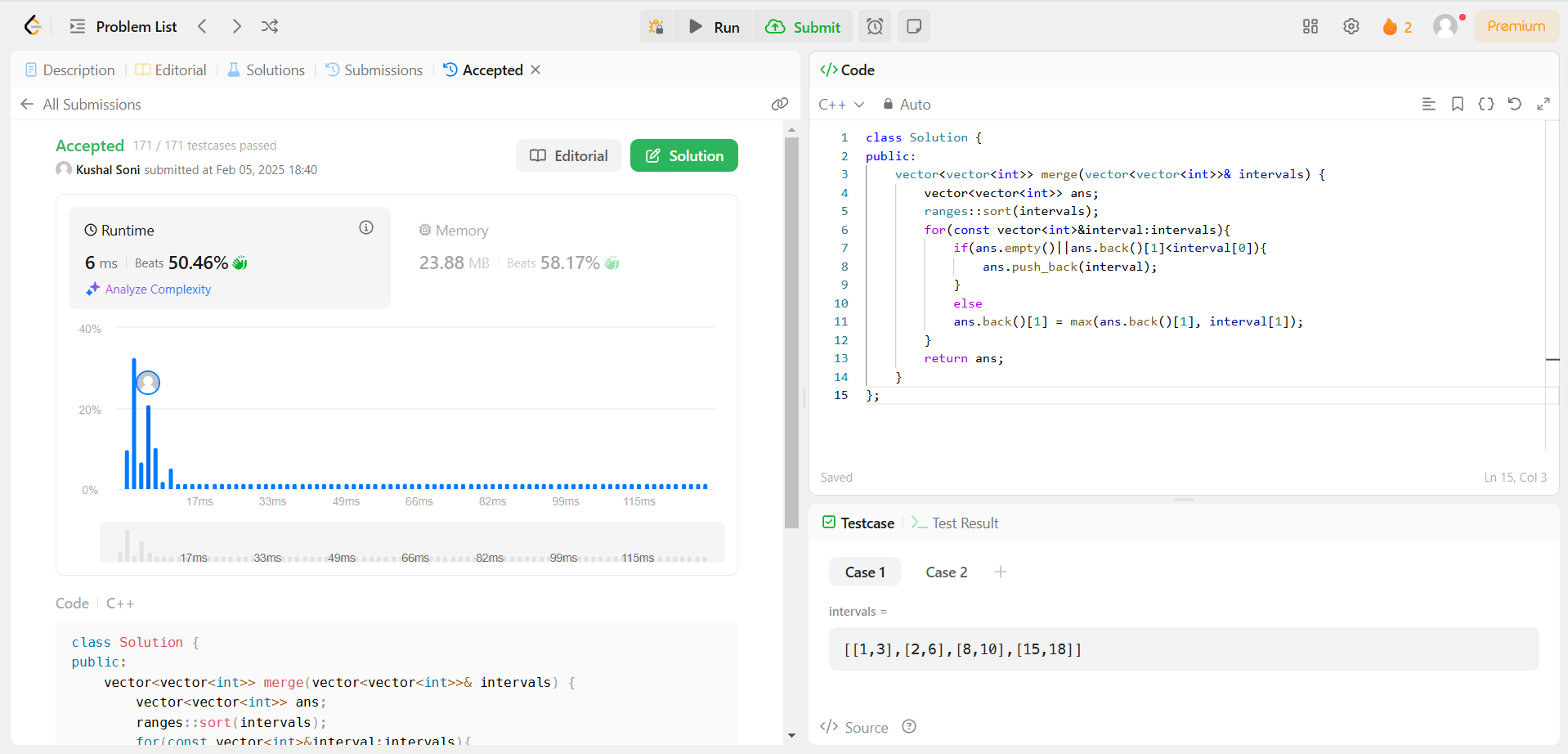
        }

        return ans;

    }

};

* **Submission:**

****

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

* **Solution Code:**

class Solution {

public:

    int pivotIndex(vector<int>& nums){

        int s=0;

        int e=nums.size()-1;

        while(s<=e){

            int mid=s+(e-s)/2;

            if(mid+1< nums.size() && nums[mid]>nums[mid+1])

                return mid;

            if(mid-1>=0 && nums[mid-1]>nums[mid])

                return mid-1;

            if(nums[s]>=nums[mid])

                e=mid-1;

            else

                s=mid+1;

        }

        return s;

    }

    int Binary(vector<int>& nums, int target, int s, int e){

        while(s<=e){

            int mid=s+(e-s)/2;

            if(nums[mid]==target)

                return mid;

            else if(nums[mid]<target)

                s=mid+1;

            else

                e=mid-1;

        }

        return -1;

    }

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

        int pivot=pivotIndex(nums);

        int ans=-1;

        if(target>=nums[0] && target<=nums[pivot])

            ans=Binary(nums, target, 0, pivot);

        if(pivot+1<nums.size() && target>=nums[pivot+1] && target<=nums[nums.size()-1])

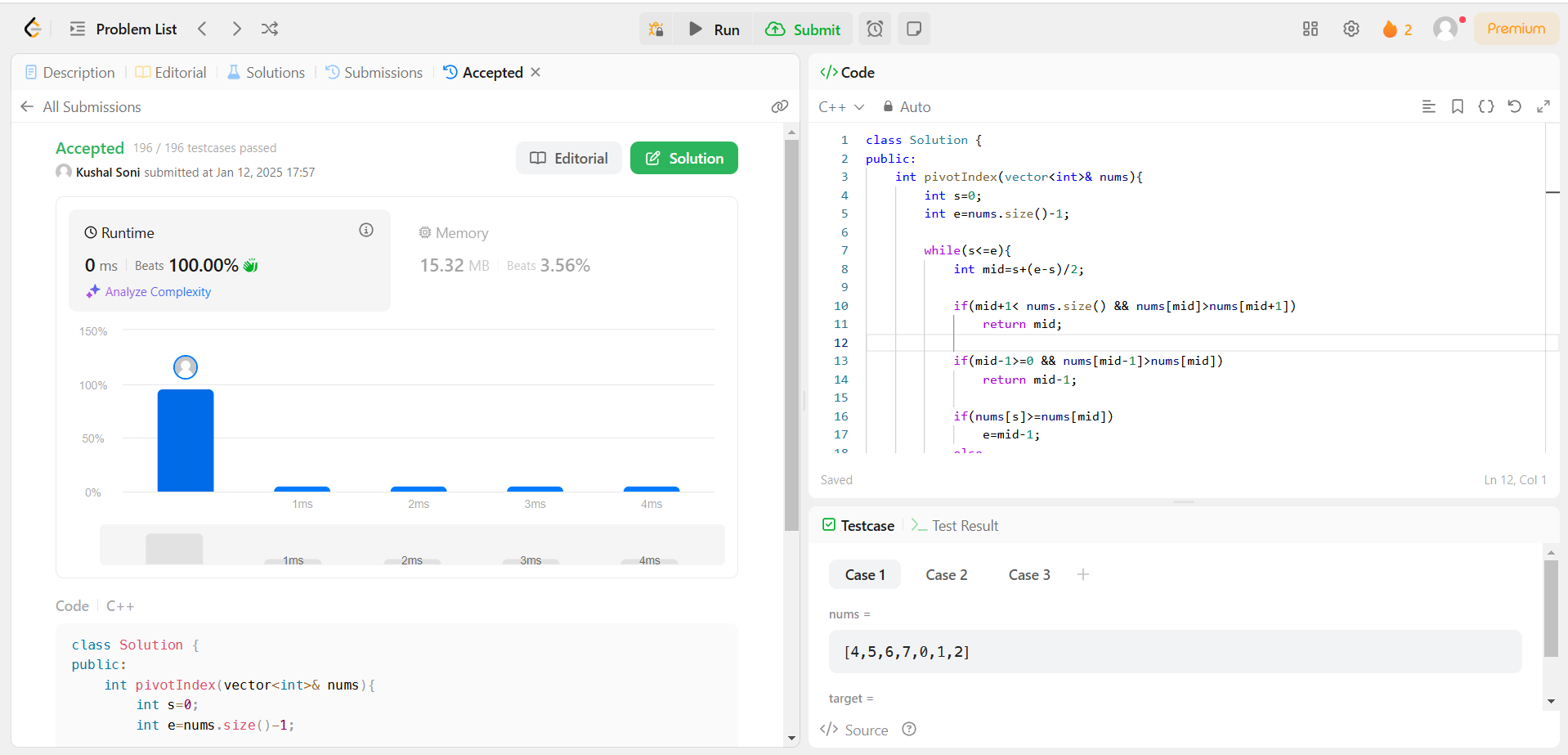
            ans=Binary(nums, target, pivot+1, nums.size()-1);

        return ans;

    }

};

* **Submission:**



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

* **Solution Code:**

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]);

        }

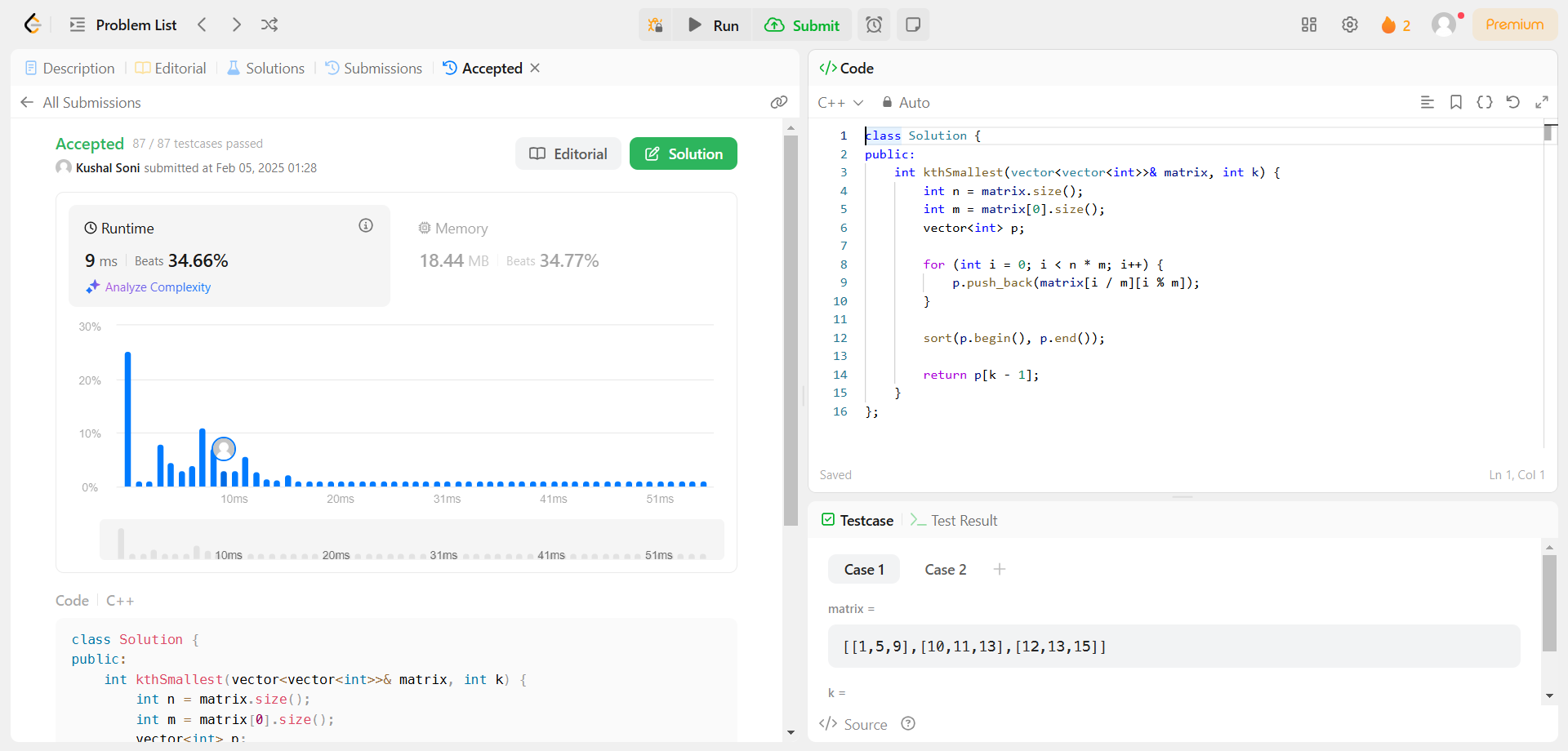
        sort(p.begin(), p.end());

        return p[k - 1];

    }

};

* **Submission:**



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

* **Solution Code:**

class Solution {

public:

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

        int n1 = nums1.size();

        int n2 = nums2.size();

        int i = 0, j = 0, m1 = 0, m2 = 0;

        for (int c = 0; c <= (n1 + n2) / 2; c++) {

            m2 = m1;

            if (i != n1 && j != n2) {

                if (nums1[i] > nums2[j]) {

                    m1 = nums2[j++];

                } else {

                    m1 = nums1[i++];

                }

            } else if (i < n1) {

                m1 = nums1[i++];

            } else {

                m1 = nums2[j++];

            }

        }

        if ((n1 + n2) % 2 == 1) {

            return static\_cast<double>(m1);

        } else {

            double ans = static\_cast<double>(m1) + static\_cast<double>(m2);

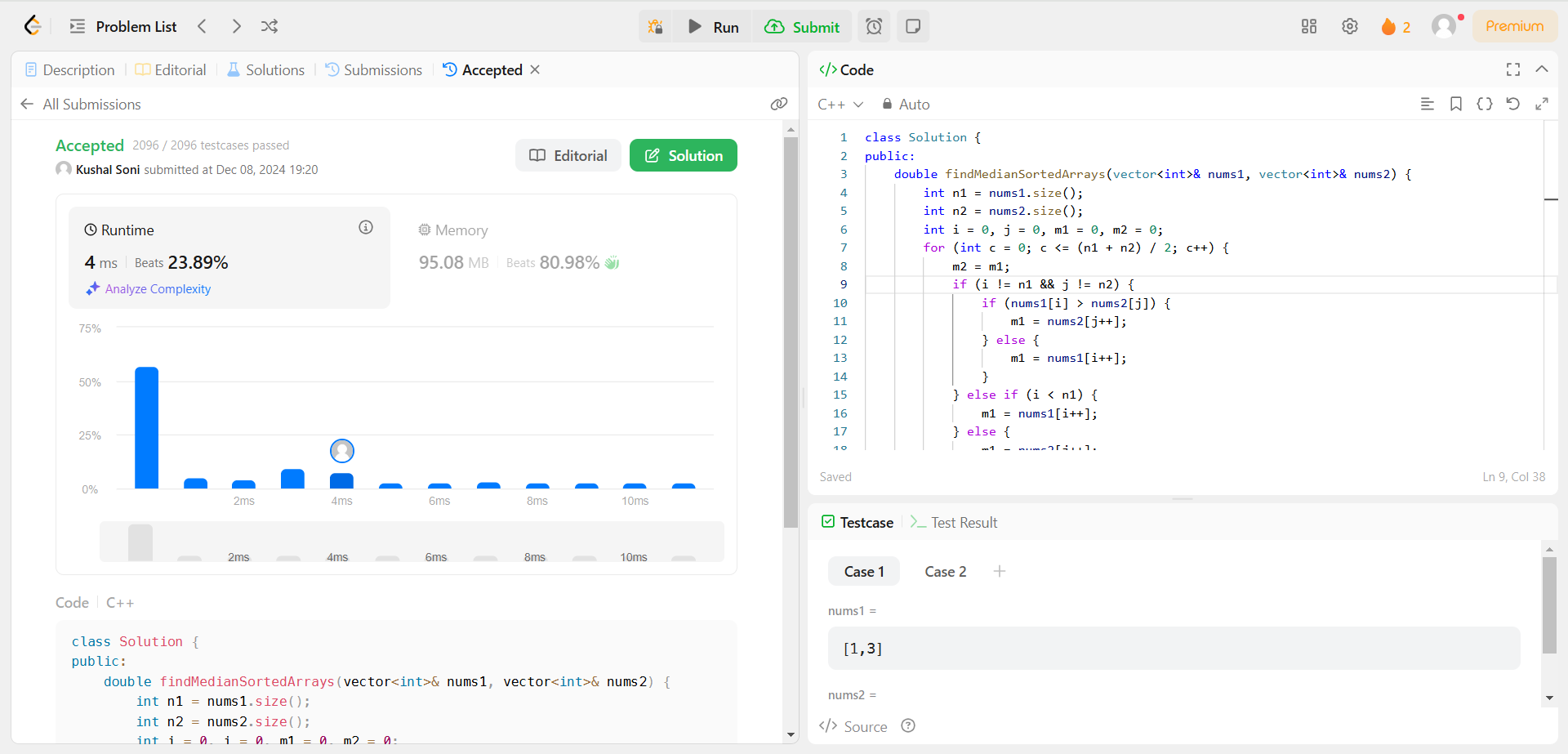
            return ans / 2.0;

        }

    }

};

* **Submission:**

****