**ASSIGNMENT-1(AP)**

**Name:** Rachit Jain **Submitted To:** Er. Vishal Kumar (E18147)

**UID:** 22BCS12152 **Section:** 22BCS\_FL\_IOT-**604-A**

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

public:

string longestNiceSubstring(string s) {

if(s.length()<=1)

{

return "";

}

unordered\_set<char>st;

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

{

st.insert(s[i]);

}

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

{

char ch=s[i];

if(st.find(tolower(ch))==st.end() || st.find(toupper(ch))==st.end())

{

string s1=longestNiceSubstring(s.substr(0,i));

string s2=longestNiceSubstring(s.substr(i+1,s.length()));

if(s1.length()>=s2.length())

{

return s1;

}

else

{

return s2;

}

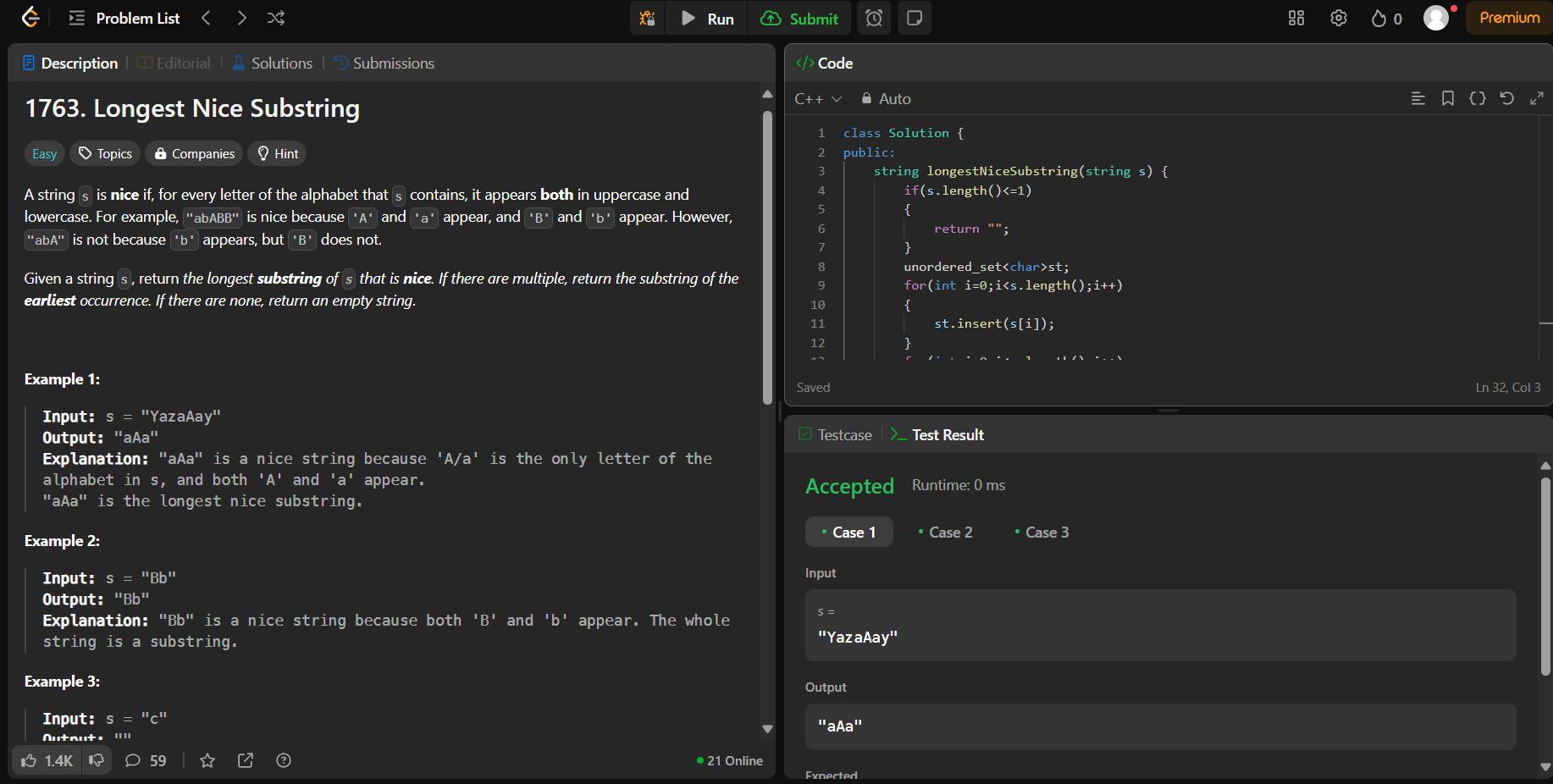
}

}

return s;

}

};

****

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

class Solution {

public:

    uint32\_t reverseBits(uint32\_t n) {

        uint32\_t ans = 0;

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

            ans <<= 1;

            ans |= (n & 1);

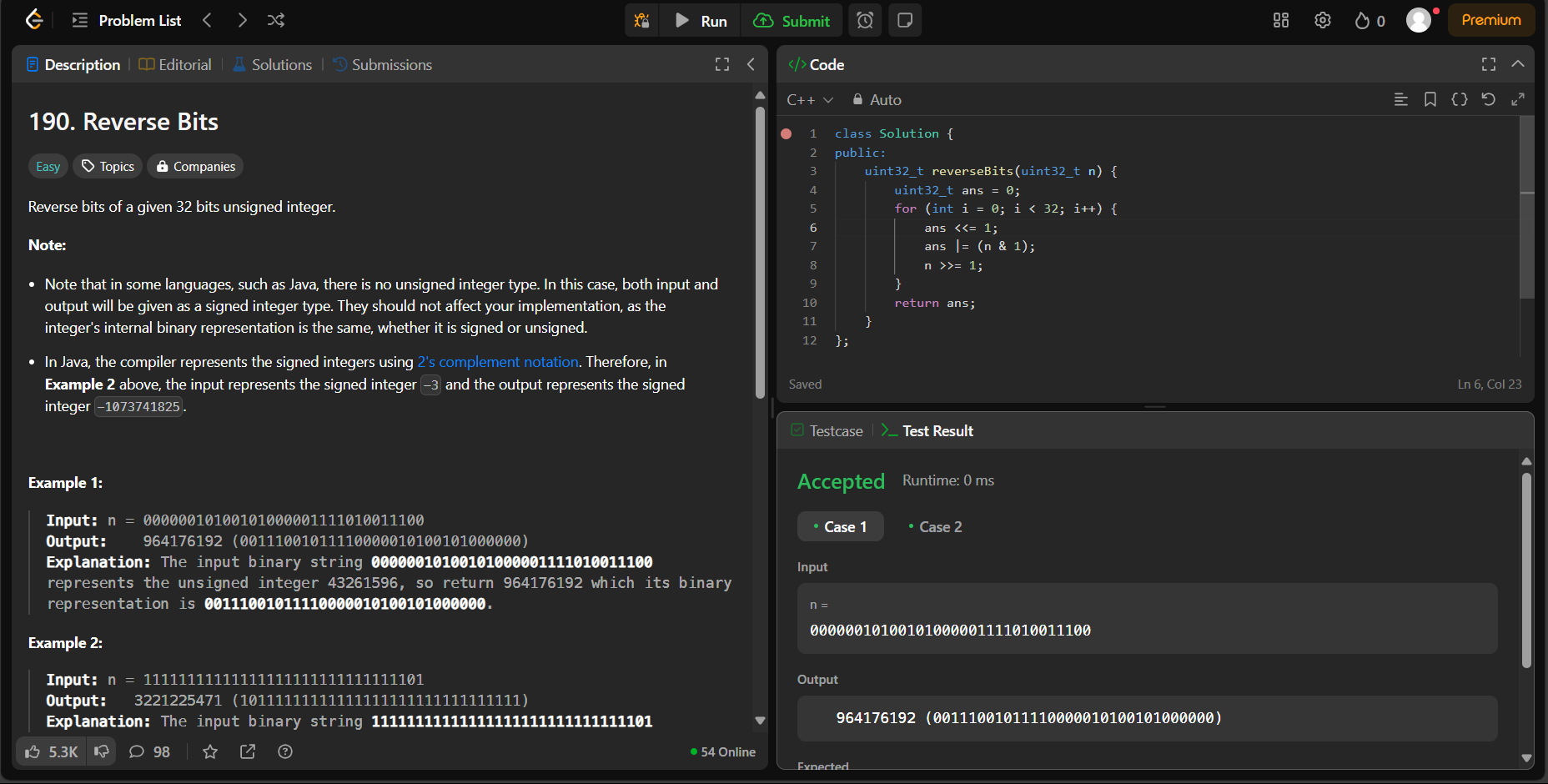
            n >>= 1;

        }

        return ans;

    }

};



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

class Solution {

public:

    int hammingWeight(int n) {

        int count = 0;

        for(int i = 31; i >= 0; i--){

            if(((n >> i) & 1) == 1)

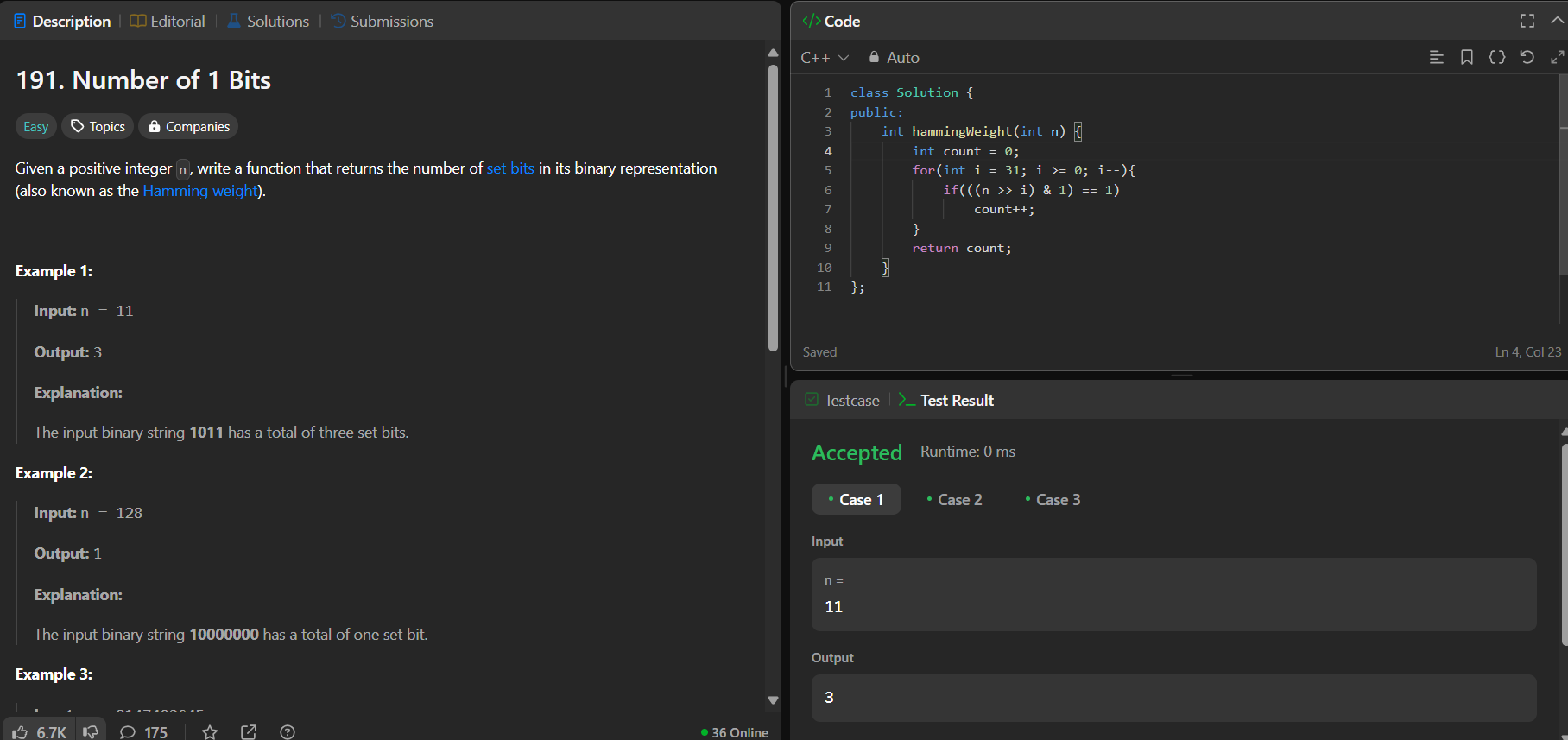
                count++;

        }

        return count;

    }

};



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

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;

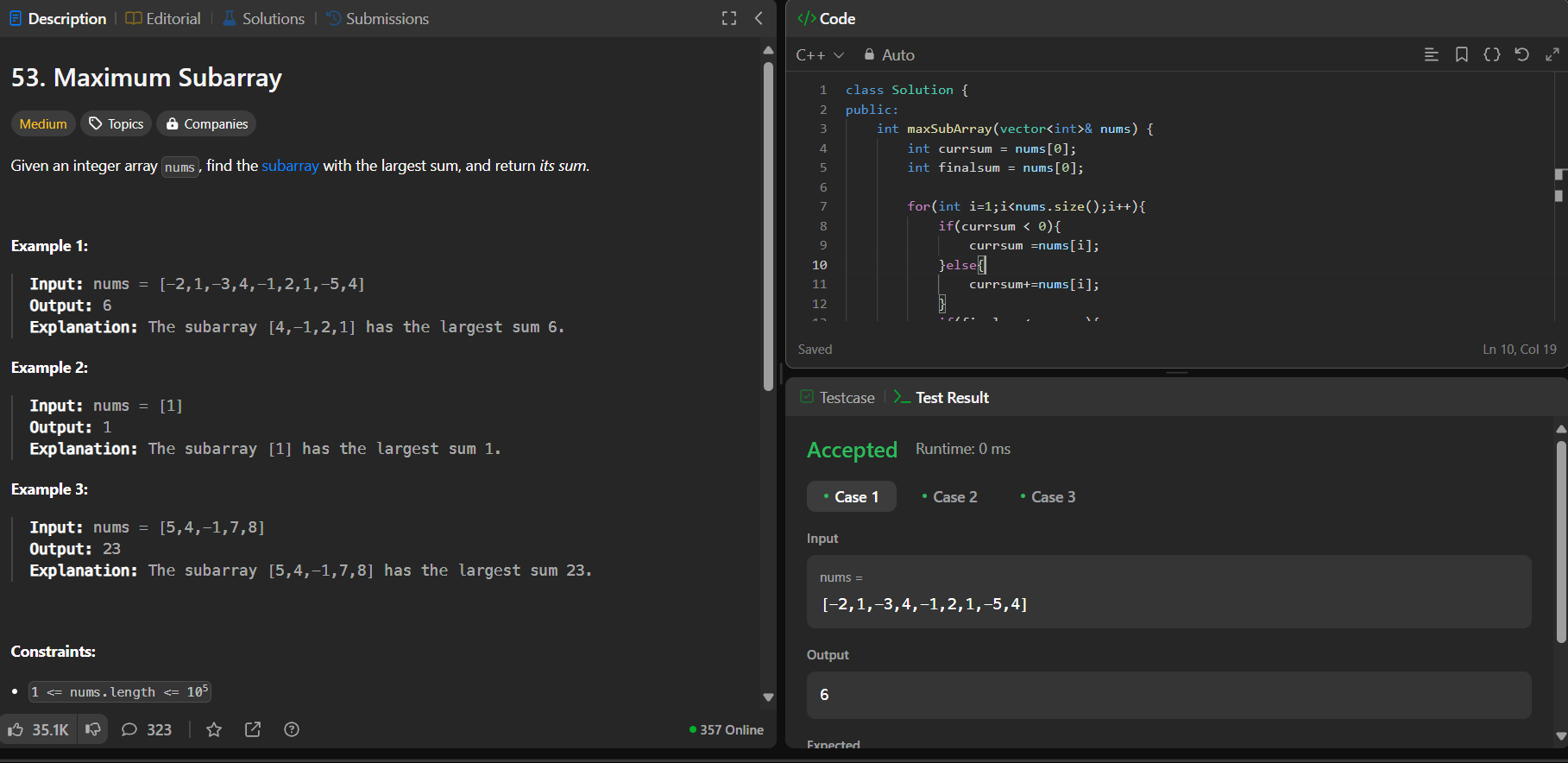
            }

        }

        return finalsum;

    }

};

****

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

class Solution {

public:

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

int cols = matrix[0].size() - 1;

int n = matrix.size() - 1;

int rows = 0;

while(rows <= n && cols >= 0){

int toCompare = matrix[rows][cols];

if(toCompare > target){

cols--;

}else if(toCompare < target){

rows++;

}else{

return true;

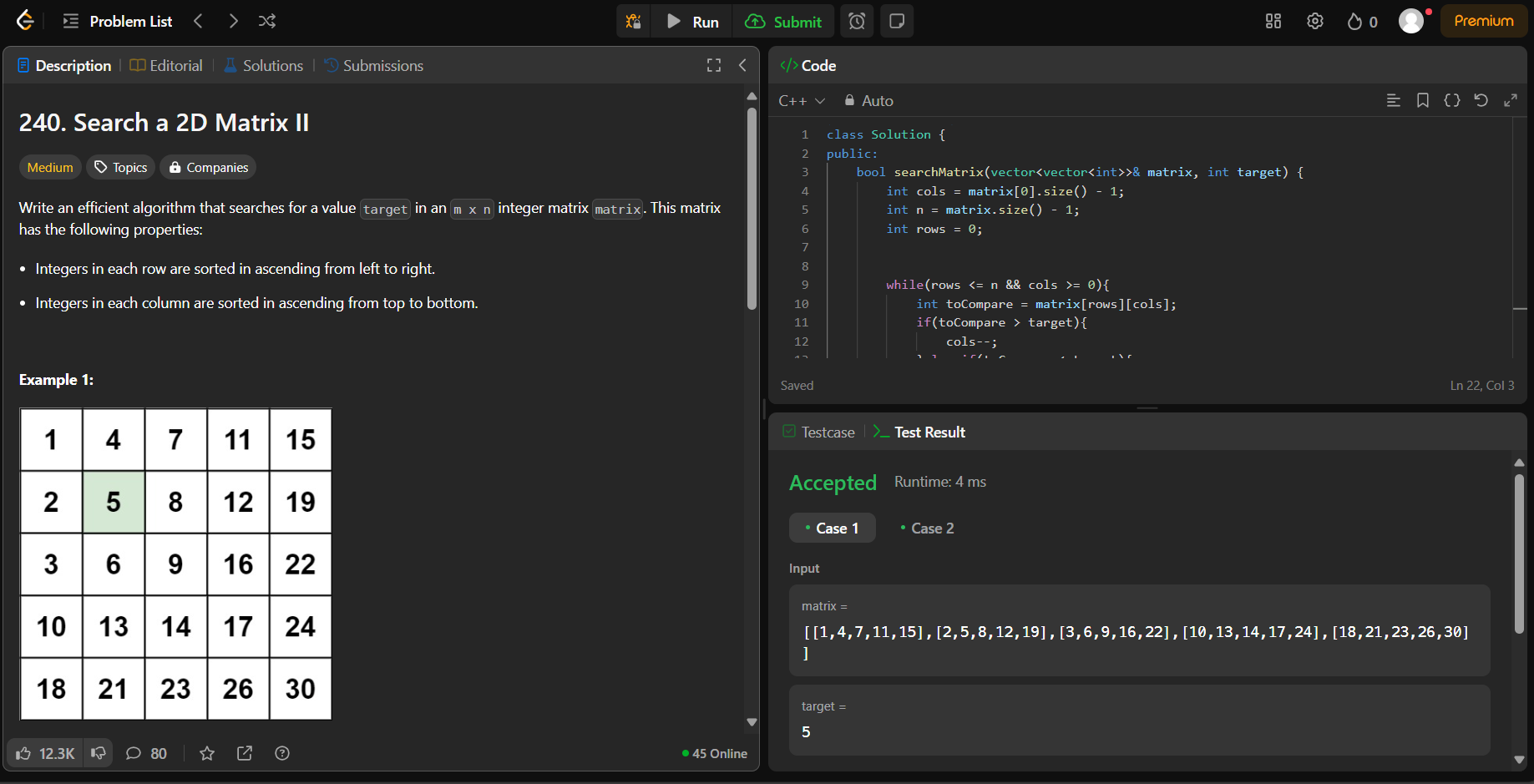
}

}

return false;

}

};

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

class Solution {

public:

const int MOD = 1337;

int pow(int a, int b) {

int result = 1;

a %= MOD; // Taking mod to prevent overflow

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

result = (result \* a) % MOD;

}

return result;

}

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

int result = 1;

for (int i = b.size() - 1; i >= 0; i--) {

result = (result \* pow(a, b[i])) % MOD;

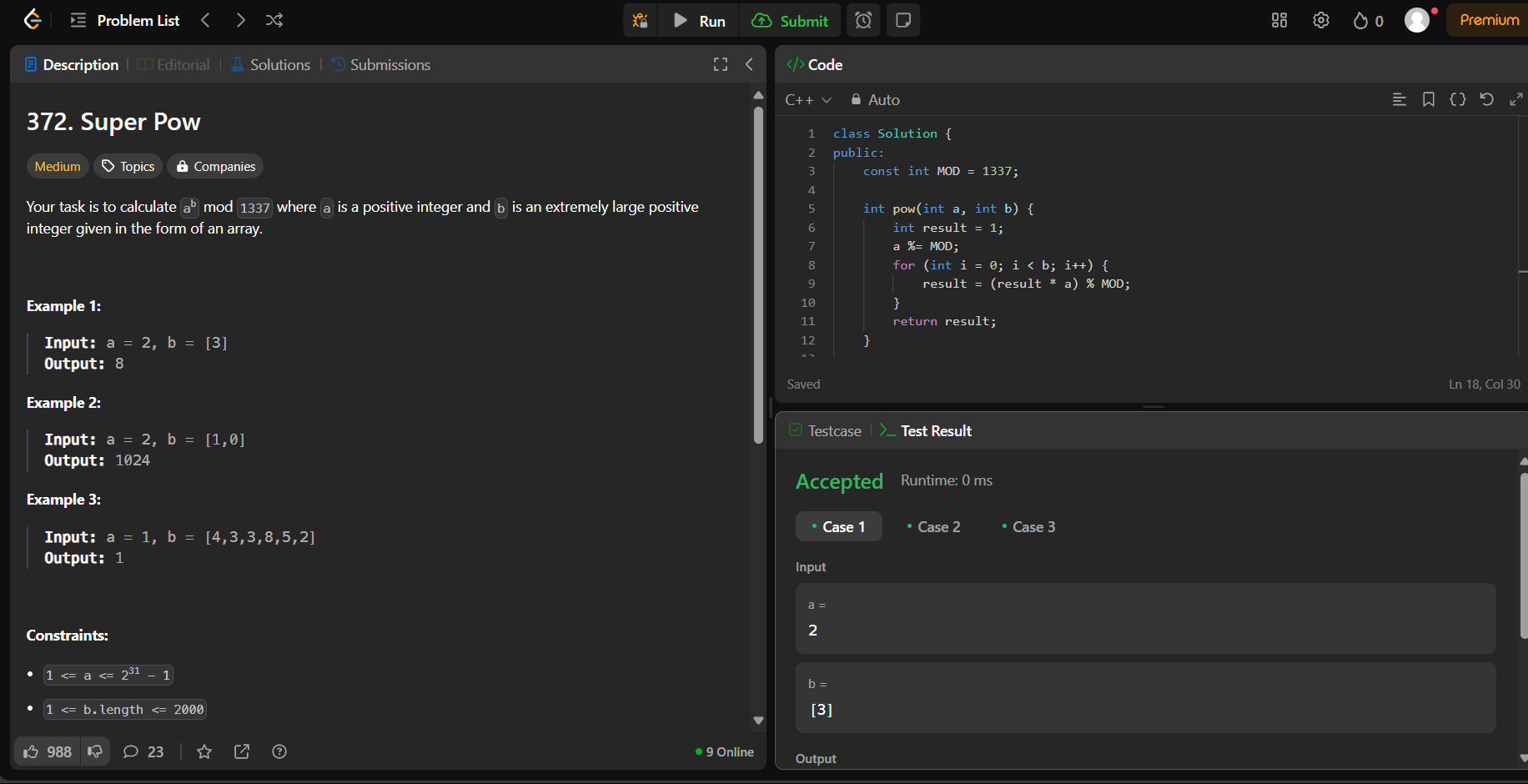
a = pow(a, 10); // Power up for the next iteration

}

return result;

}

};

**  
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 (auto i : res) if (2 \* i - 1 <= n) tmp.push\_back(2 \* i - 1);

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

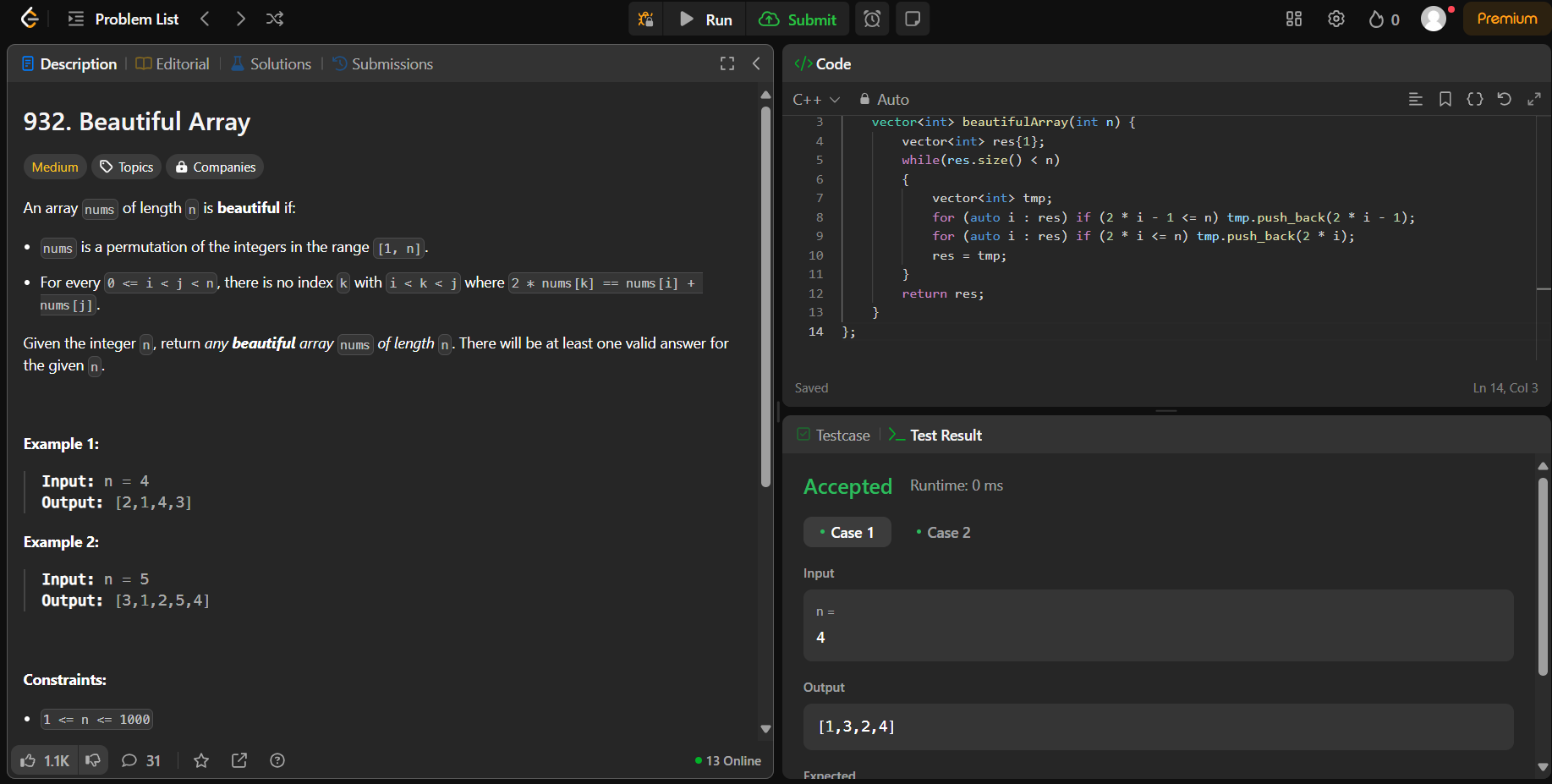
res = tmp;

}

return res;

}

};

****

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

#include <vector>

class Solution {

public:

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

int i = m - 1, j = n - 1, k = m + n - 1;

while (i >= 0 && j >= 0) {

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

nums1[k--] = nums1[i--];

} else {

nums1[k--] = nums2[j--];

}

}

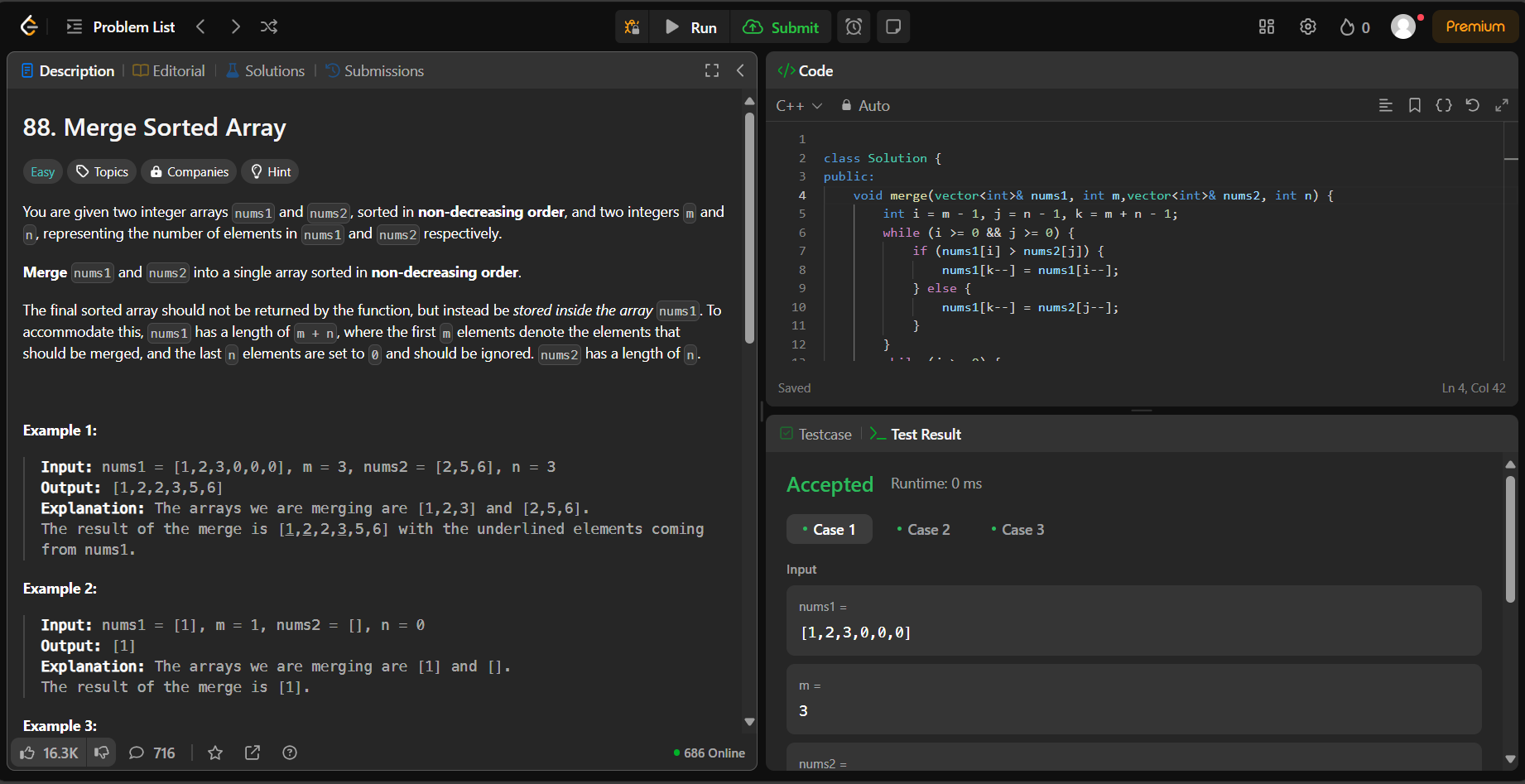
while (j >= 0) {

nums1[k--] = nums2[j--];

}

}

};

****

**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;

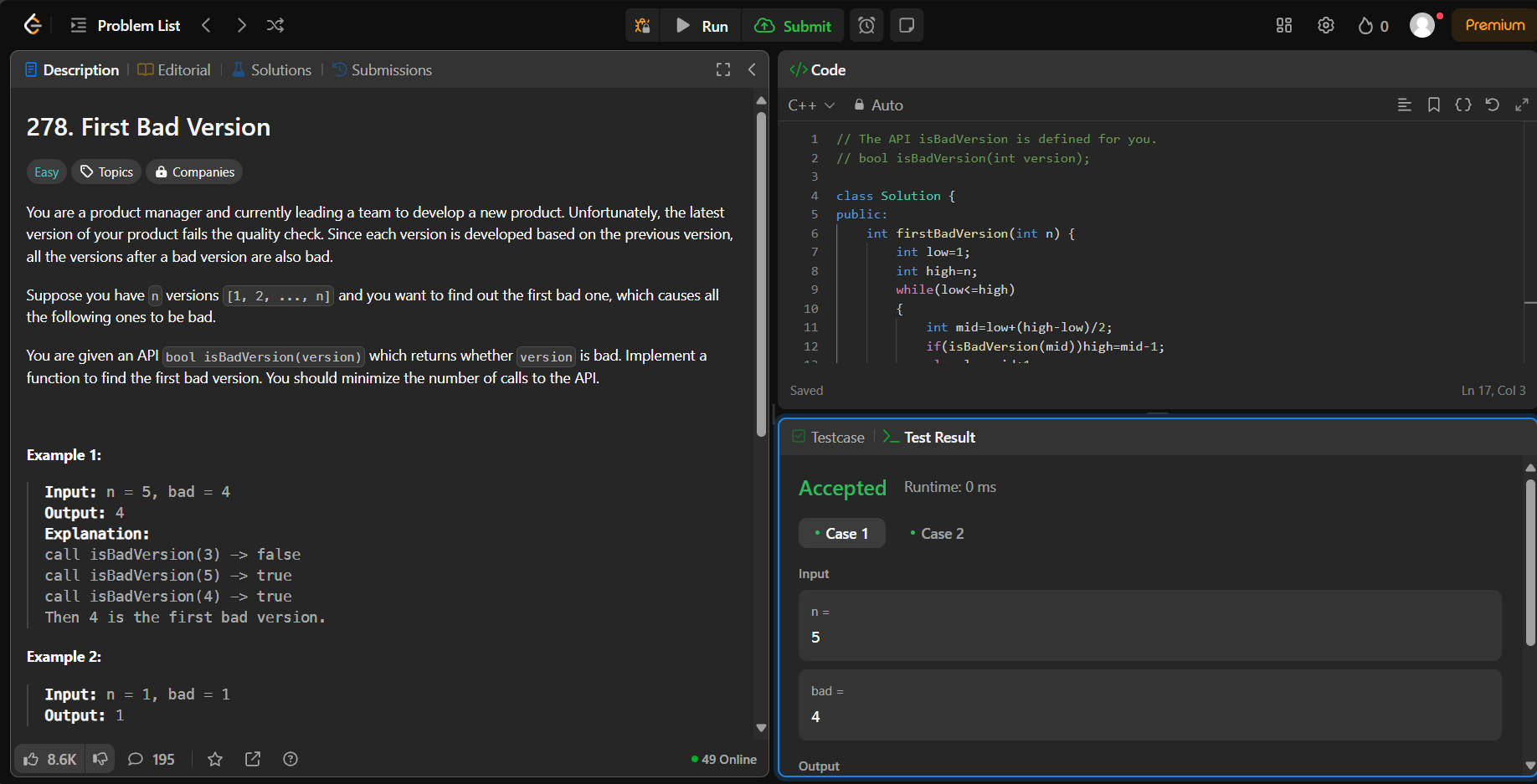
else low=mid+1;

}

return low;

}

};

****

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

class Solution {

public:

void sortColors(vector<int>& nums) {

int zeros = 0, ones = 0, n = nums.size();

for(int num : nums) {

if(num == 0) zeros++;

else if(num == 1) ones++;

}

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

nums[i] = 0;

}

for(int i = zeros; i < zeros + ones; ++i) {

nums[i] = 1;

}

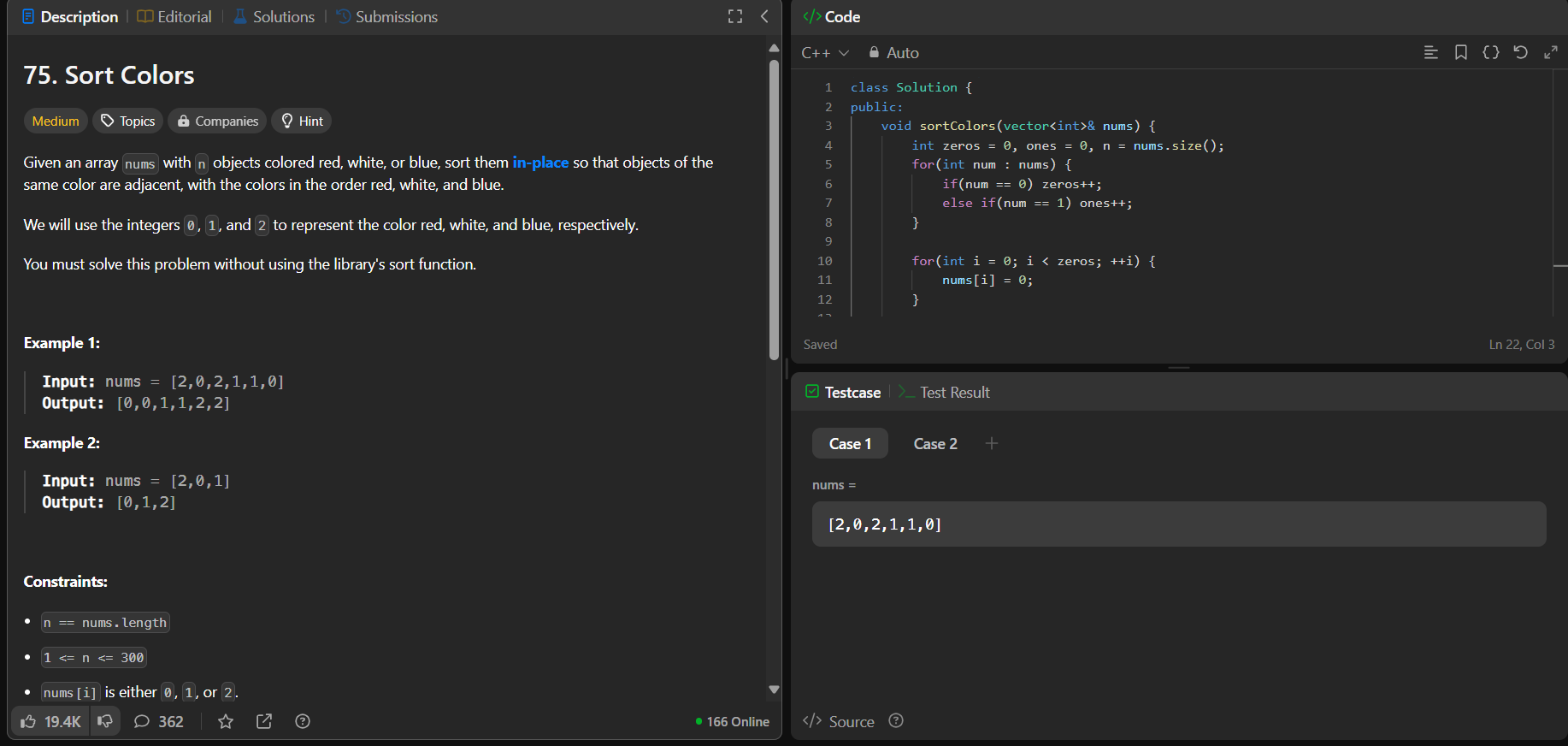
for(int i = zeros + ones; i < n; ++i) {

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) {

vector<int> res;

unordered\_map<int, int> temp;

for (int num : nums) {

temp[num]++;

}

priority\_queue<pair<int, int>> res1;

for (auto& pair : temp) {

res1.push({pair.second, pair.first});

}

while (k > 0) {

pair<int, int> curr = res1.top();

res.push\_back(curr.second);

res1.pop();

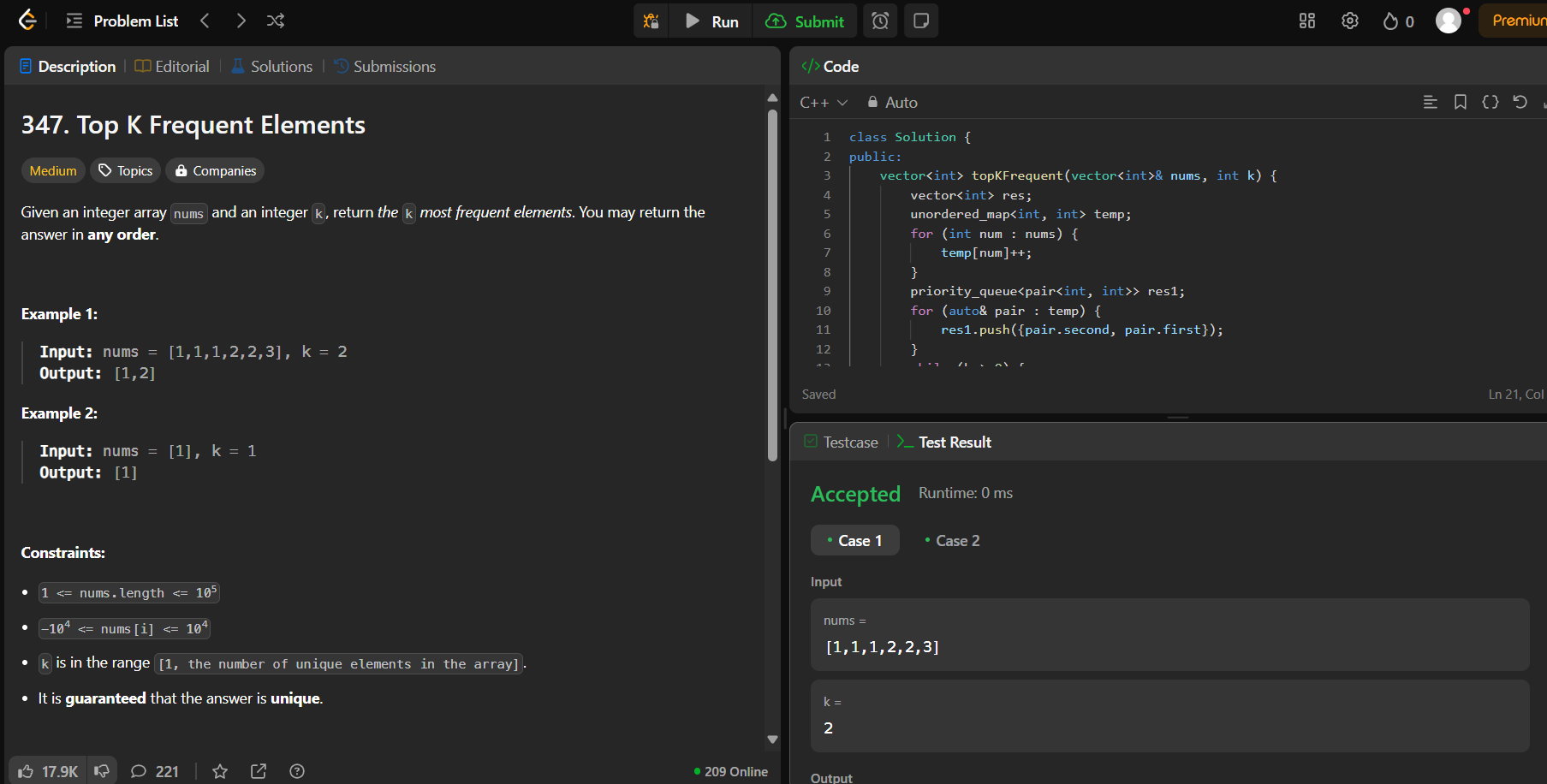
k--;

}

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

        priority\_queue<int> pq ;

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

            pq.push(nums[i]);

        }

        k--;

        while(k>0 ){

            pq.pop();

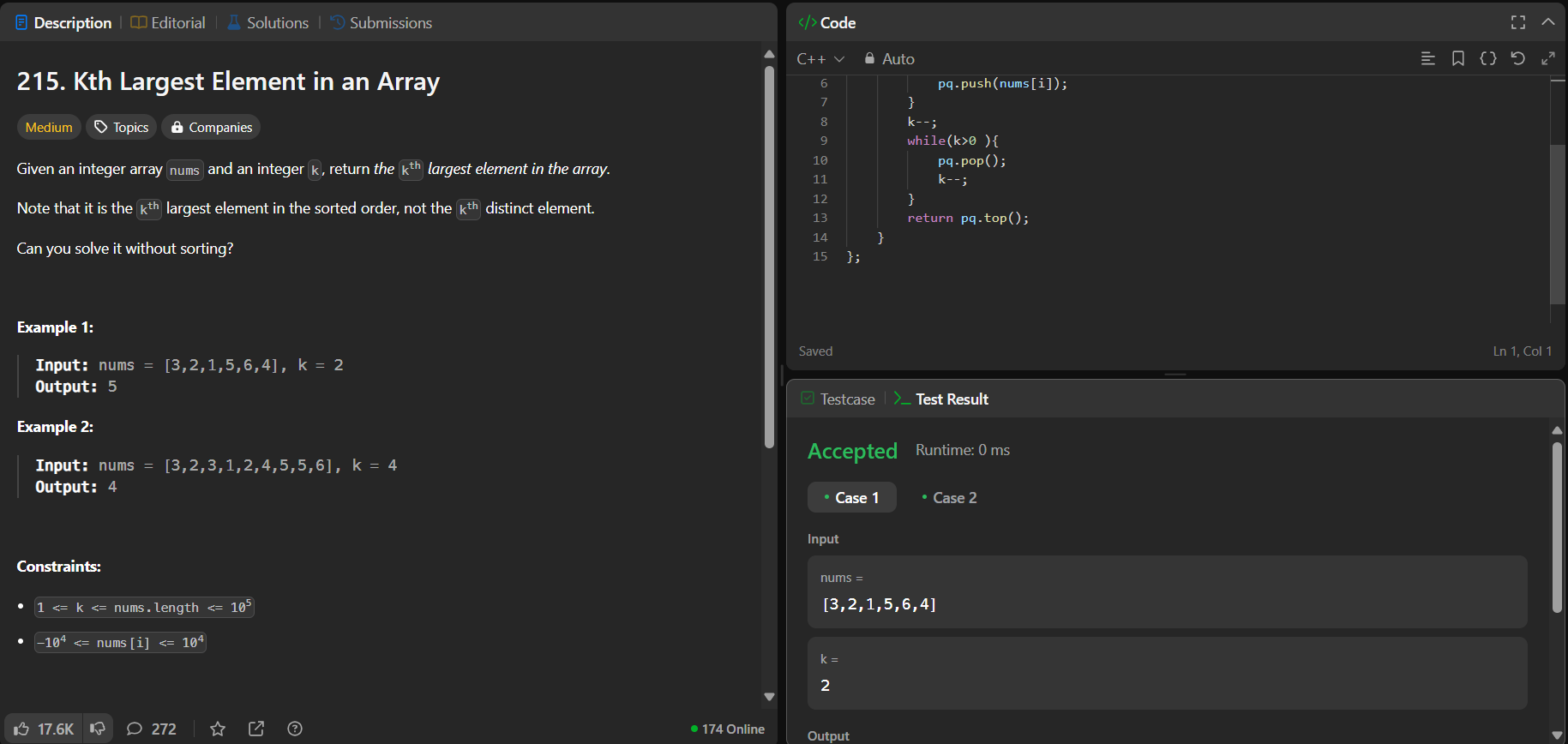
            k--;

        }

        return pq.top();

    }

};

****

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

class Solution {

public:

int findPeakElement(vector<int>& nums) {

int n = nums.size();

int r = n-1;

int l = 0;

while(r>l){

int mid = (r+l)/2;

if(nums[mid] > nums[mid+1]) r = mid;

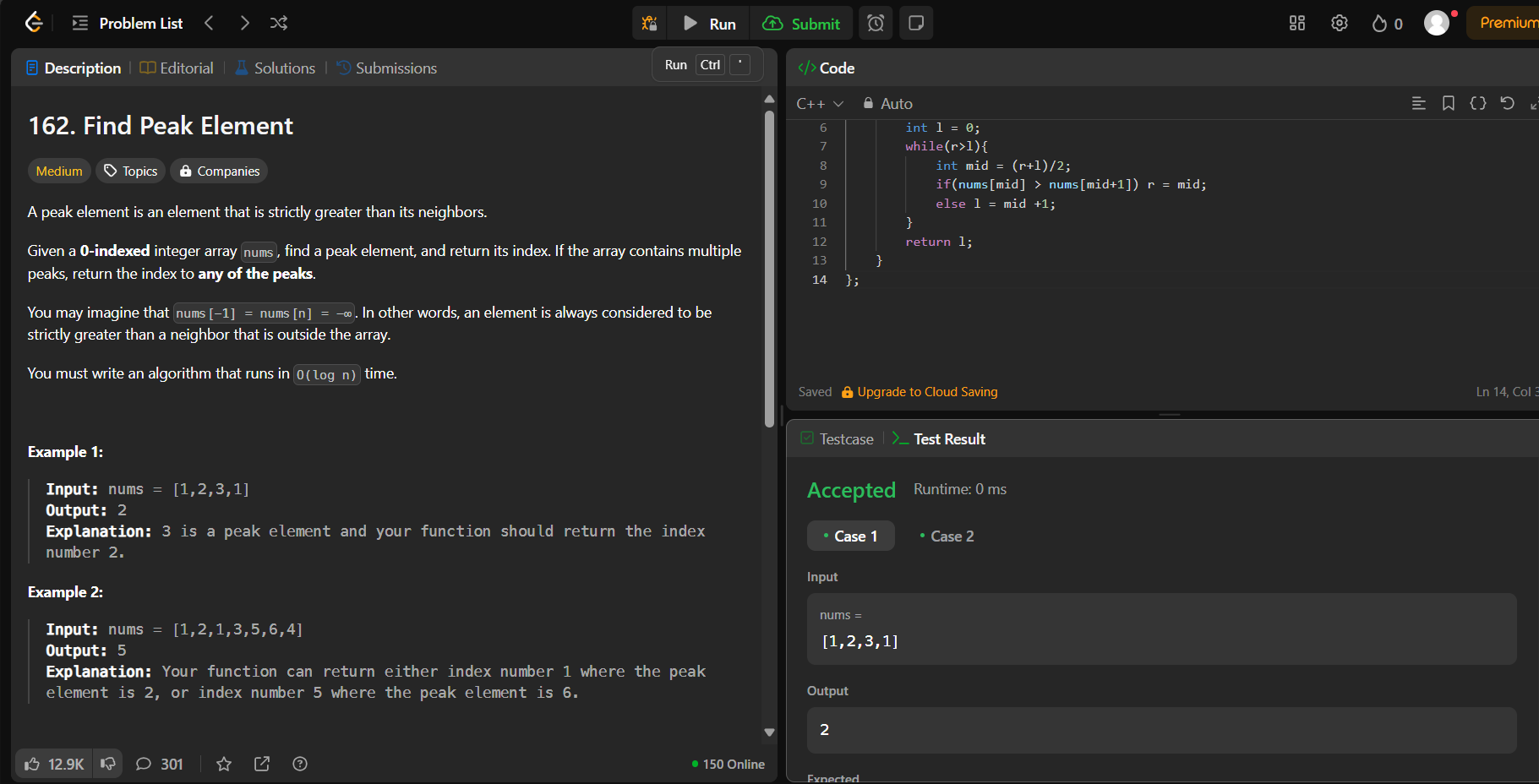
else l = mid +1;

}

return l;

}

};

****

**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 start=0;

int end=nums.size()-1;

while(start<=end){

int mid=start+(end-start)/2;

if(nums[mid]==target){

return mid;

}

if(nums[start]<=nums[mid]){

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

end= mid-1;

}

else{

start=mid+1;

}

}

else{

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

start=mid+1;

}

else{

end= mid-1;

}

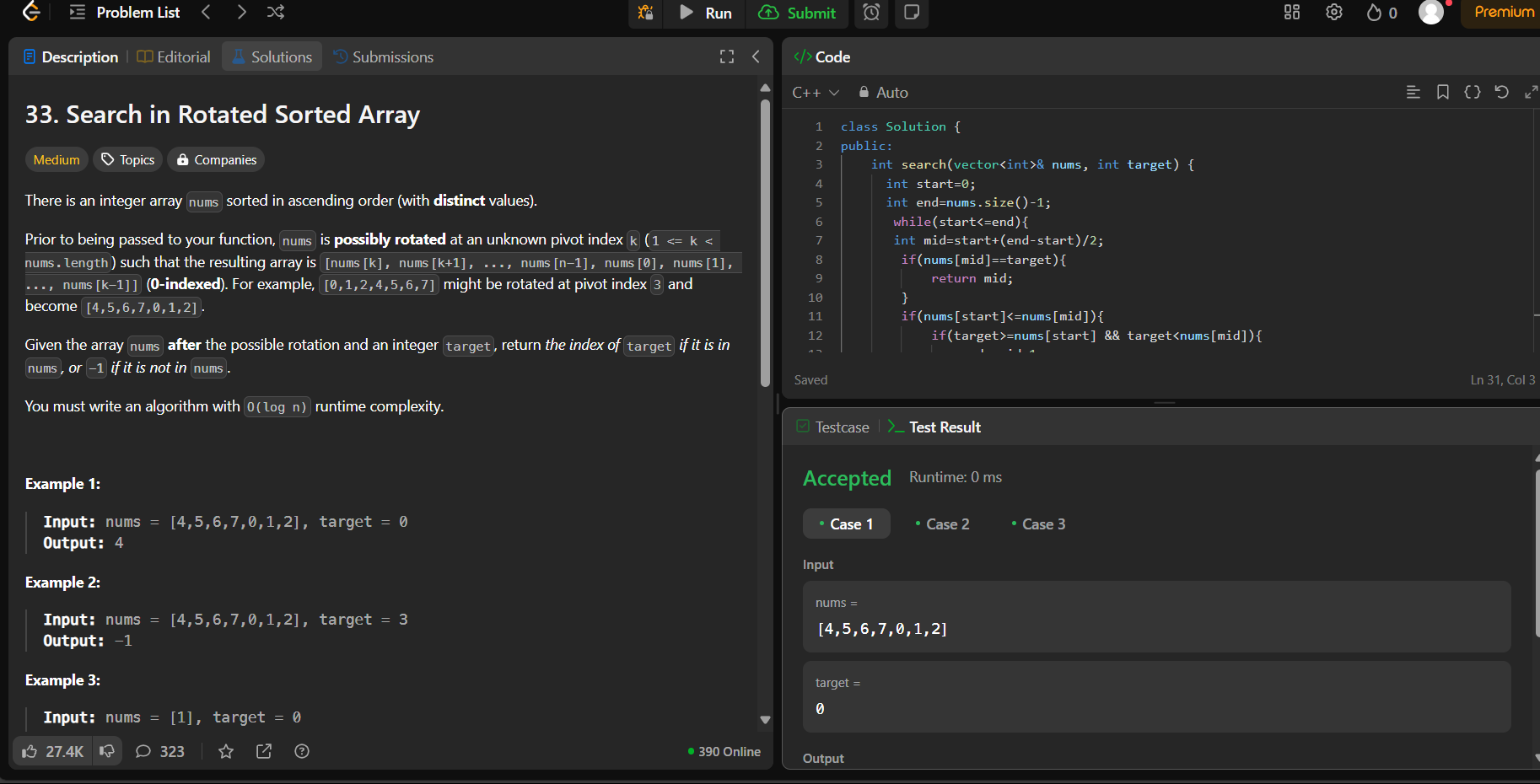
}

}

return -1;

}

};

****

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

class Solution {

public:

void wiggleSort(vector<int>& nums) {

vector<int> tmp=nums;

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

int x=nums.size()/2;

int y=nums.size()-x;

int i=1,k=nums.size()-1;

while(x--){

nums[i]=tmp[k--];

i+=2;

}

i=0;

while(y--){

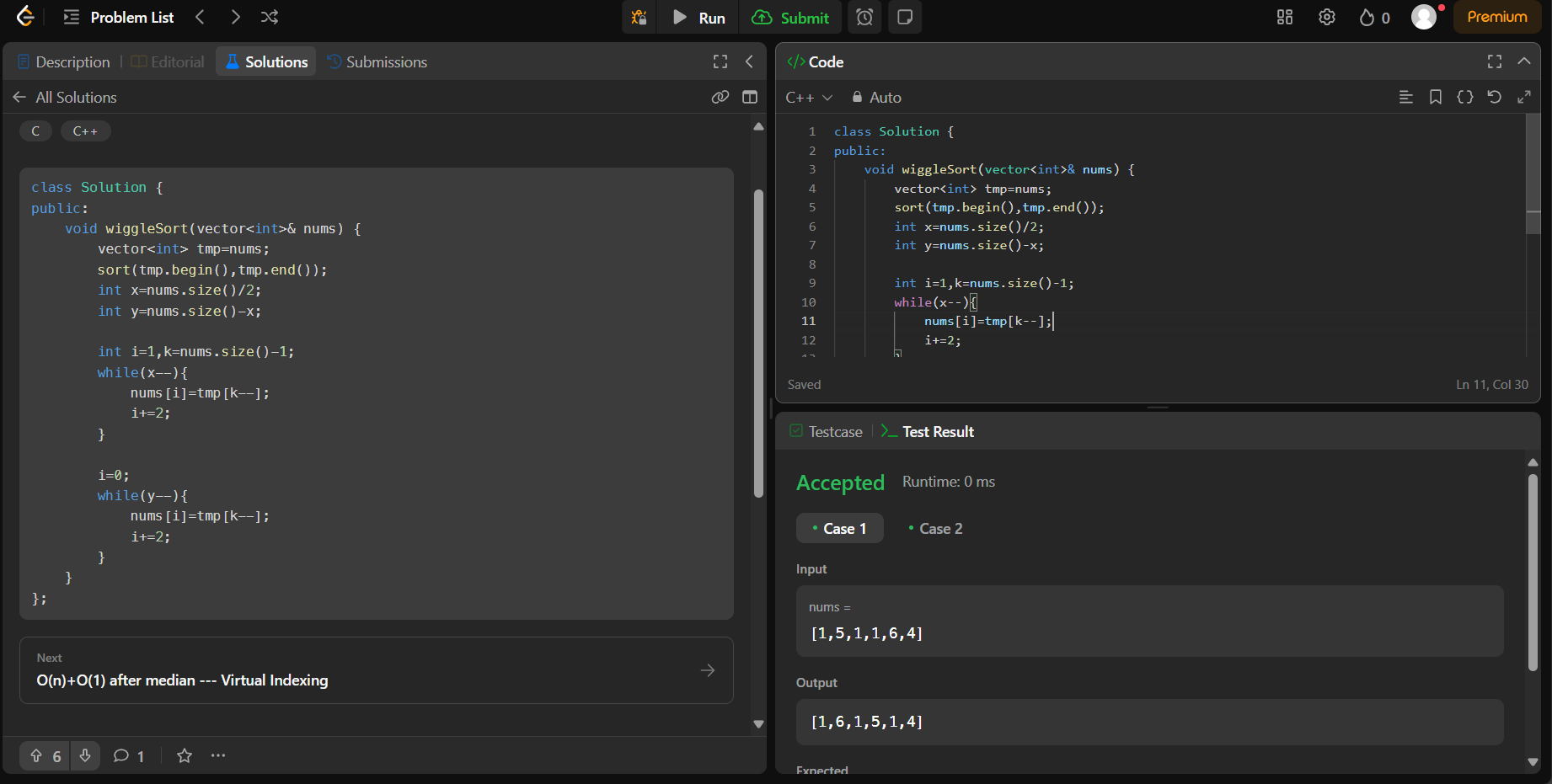
nums[i]=tmp[k--];

i+=2;

}

}

};

****

**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) {

vector<int> result;

for (const auto& row : matrix) {

for (const auto& element : row)

result.push\_back(element);

}

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

int size=result.size();

if(size<k) return -1;

return result[k-1];

}

};

