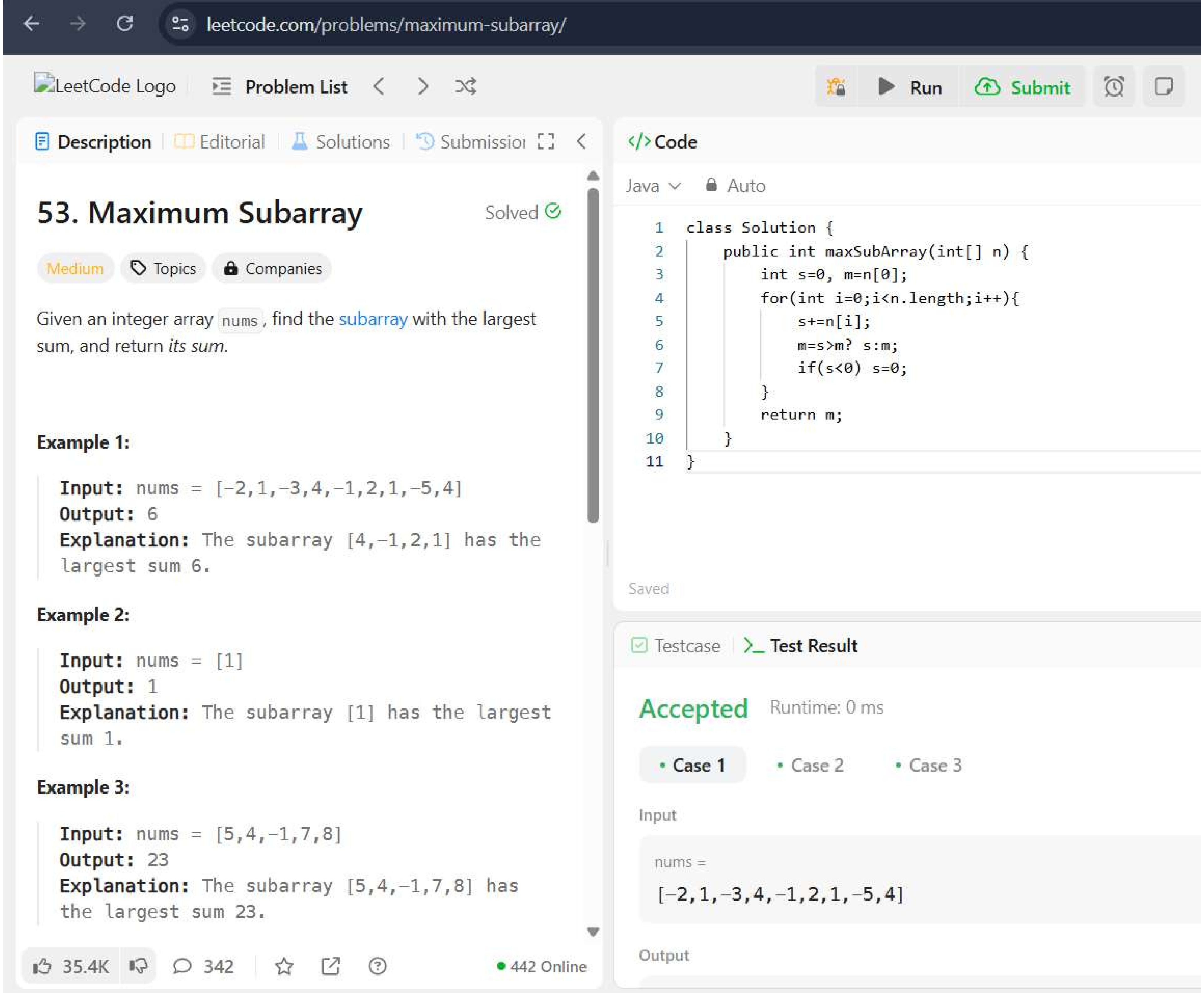


53. Maximum Subarray

Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*.

Example 1:

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the
largest sum 6.
```

Example 2:

```
Input: nums = [1]
Output: 1
Explanation: The subarray [1] has the largest
sum 1.
```

Example 3:

```
Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has
the largest sum 23.
```

```java
class Solution {
    public int maxSubArray(int[] n) {
        int s=0, m=n[0];
        for(int i=0;i<n.length;i++){
            s+=n[i];
            m=s>m? s:m;
            if(s<0) s=0;
        }
        return m;
    }
}
```

Saved

Accepted Runtime: 0 ms

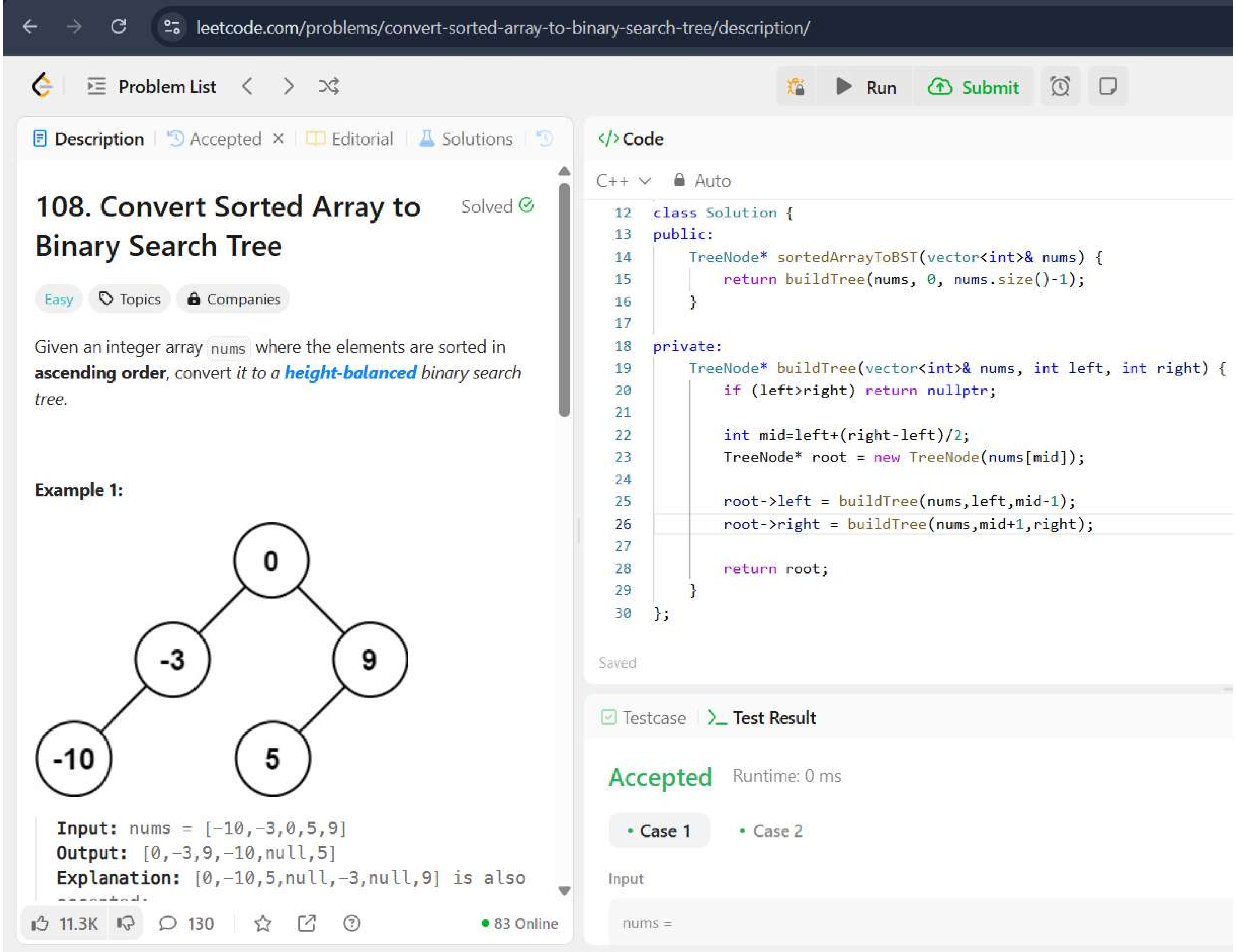
• Case 1 • Case 2 • Case 3

Input

nums =

[-2,1,-3,4,-1,2,1,-5,4]

Output



108. Convert Sorted Array to Binary Search Tree

Solved ✓

Given an integer array `nums` where the elements are sorted in **ascending order**, convert *it to a* ***height-balanced*** *binary search tree.*

Example 1:

```
Input: nums = [-10,-3,0,5,9]
Output: [0,-3,9,-10,null,5]
Explanation: [0,-10,5,null,-3,null,9] is also
```

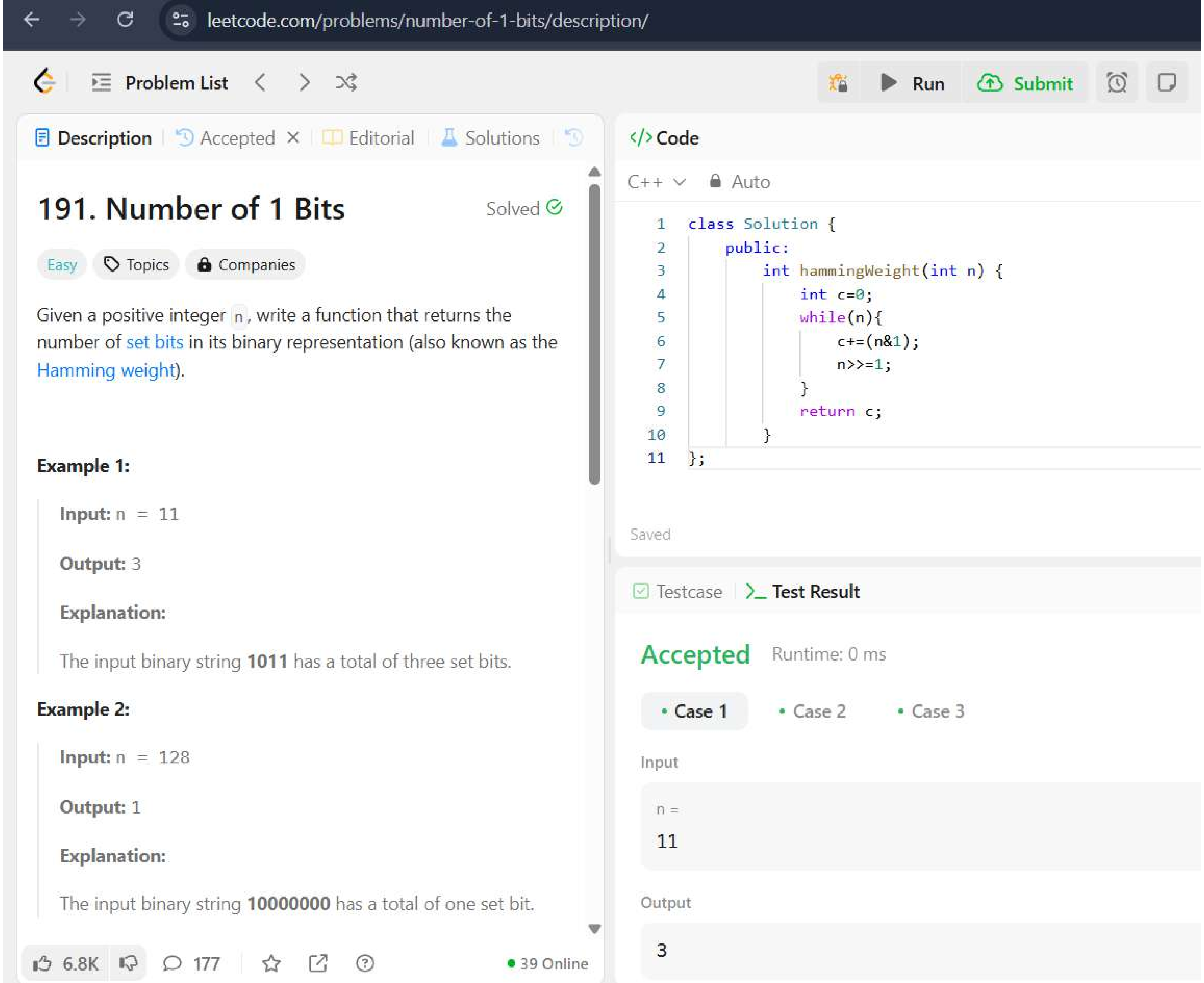
```cpp
class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        return buildTree(nums, 0, nums.size()-1);
    }

private:
    TreeNode* buildTree(vector<int>& nums, int left, int right) {
        if (left>right) return nullptr;

        int mid=left+(right-left)/2;
        TreeNode* root = new TreeNode(nums[mid]);

        root->left = buildTree(nums,left,mid-1);
        root->right = buildTree(nums,mid+1,right);

        return root;
    }
};
```

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =



191. Number of 1 Bits

Solved

Easy · Topics · Companies

Given a positive integer `n`, write a function that returns the number of set bits in its binary representation (also known as the Hamming weight).

Example 1:

Input: n = 11

Output: 3

Explanation:

The input binary string **1011** has a total of three set bits.

Example 2:

Input: n = 128

Output: 1

Explanation:

The input binary string **10000000** has a total of one set bit.

6.8K · 177 · 39 Online

```cpp
class Solution {
    public:
        int hammingWeight(int n) {
            int c=0;
            while(n){
                c+=(n&1);
                n>>=1;
            }
            return c;
        }
};
```

Accepted   Runtime: 0 ms

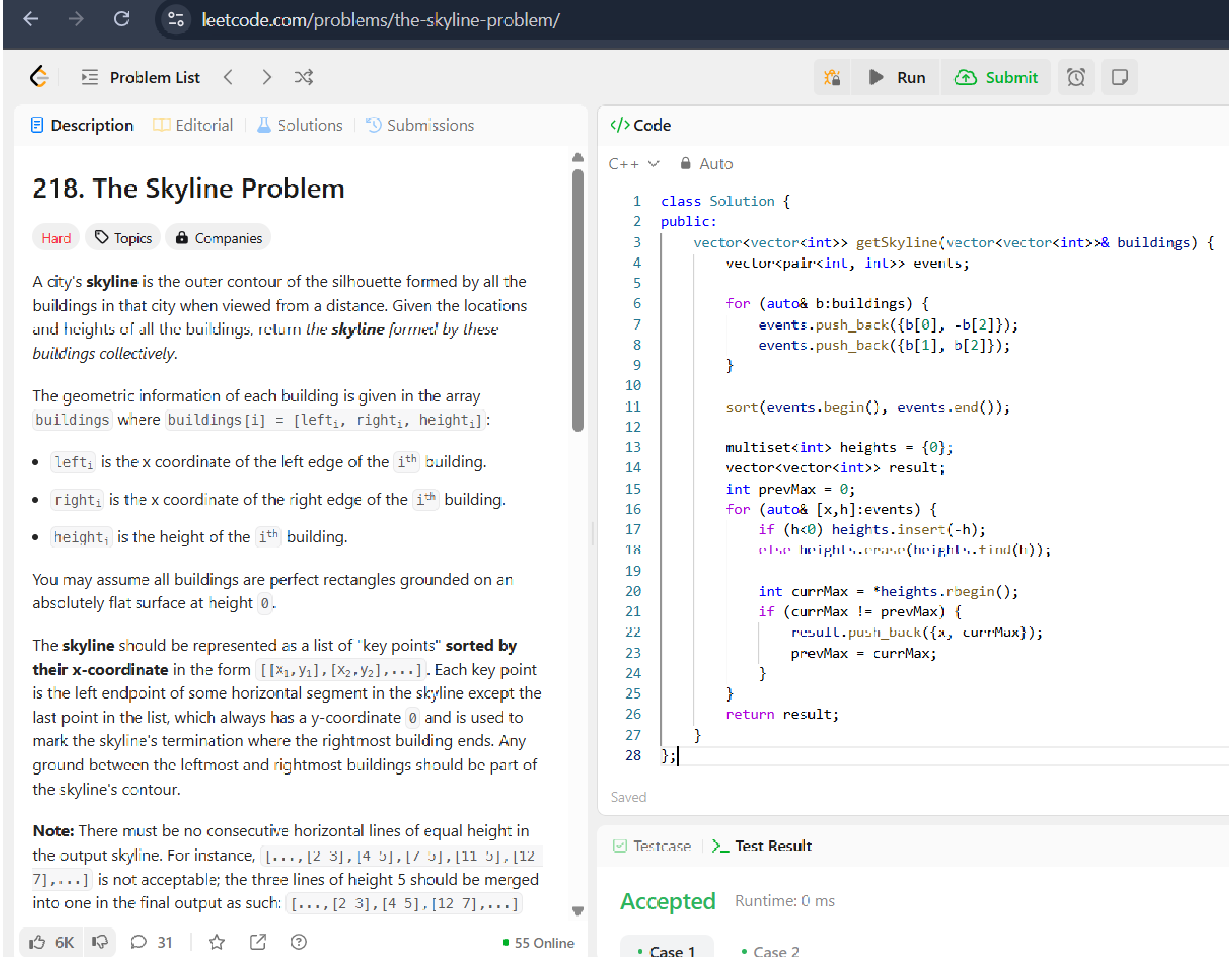
• Case 1   • Case 2   • Case 3

Input

```
n =
11
```

Output

```
3
```



218. The Skyline Problem

Hard Topics Companies

A city's **skyline** is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return *the **skyline** formed by these buildings collectively*.

The geometric information of each building is given in the array `buildings` where `buildings[i] = [left`$_i$`, right`$_i$`, height`$_i$`]`:

- `left`$_i$ is the x coordinate of the left edge of the `i`th building.
- `right`$_i$ is the x coordinate of the right edge of the `i`th building.
- `height`$_i$ is the height of the `i`th building.

You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height `0`.

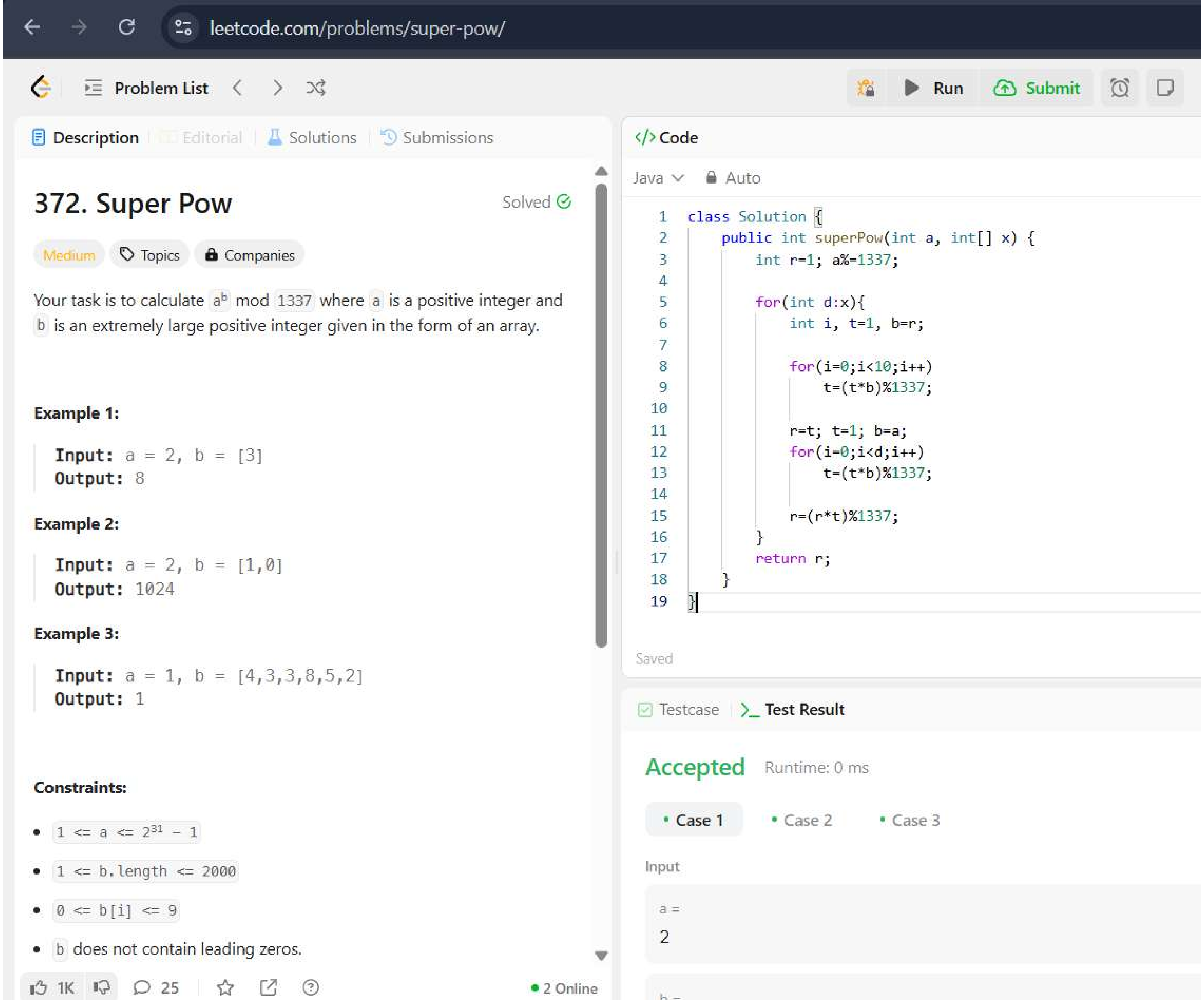
The **skyline** should be represented as a list of "key points" **sorted by their x-coordinate** in the form `[[x`$_1$`,y`$_1$`],[x`$_2$`,y`$_2$`],...]`. Each key point is the left endpoint of some horizontal segment in the skyline except the last point in the list, which always has a y-coordinate `0` and is used to mark the skyline's termination where the rightmost building ends. Any ground between the leftmost and rightmost buildings should be part of the skyline's contour.

Note: There must be no consecutive horizontal lines of equal height in the output skyline. For instance, `[...,[2 3],[4 5],[7 5],[11 5],[12 7],...]` is not acceptable; the three lines of height 5 should be merged into one in the final output as such: `[...,[2 3],[4 5],[12 7],...]`

```cpp
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;

        for (auto& b:buildings) {
            events.push_back({b[0], -b[2]});
            events.push_back({b[1], b[2]});
        }

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

        multiset<int> heights = {0};
        vector<vector<int>> result;
        int prevMax = 0;
        for (auto& [x,h]:events) {
            if (h<0) heights.insert(-h);
            else heights.erase(heights.find(h));

            int currMax = *heights.rbegin();
            if (currMax != prevMax) {
                result.push_back({x, currMax});
                prevMax = currMax;
            }
        }
        return result;
    }
};
```

Accepted Runtime: 0 ms

Case 1 Case 2



372. Super Pow

Solved ✓

Medium | Topics | Companies

Your task is to calculate a^b mod 1337 where a is a positive integer and b is an extremely large positive integer given in the form of an array.

Example 1:

```
Input: a = 2, b = [3]
Output: 8
```

Example 2:

```
Input: a = 2, b = [1,0]
Output: 1024
```

Example 3:

```
Input: a = 1, b = [4,3,3,8,5,2]
Output: 1
```

Constraints:

- `1 <= a <=` $2^{31} - 1$
- `1 <= b.length <= 2000`
- `0 <= b[i] <= 9`
- b does not contain leading zeros.

1K | 25 | 2 Online

```java
class Solution {
    public int superPow(int a, int[] x) {
        int r=1; a%=1337;

        for(int d:x){
            int i, t=1, b=r;

            for(i=0;i<10;i++)
                t=(t*b)%1337;

            r=t; t=1; b=a;
            for(i=0;i<d;i++)
                t=(t*b)%1337;

            r=(r*t)%1337;
        }
        return r;
    }
}
```

Saved

Accepted Runtime: 0 ms

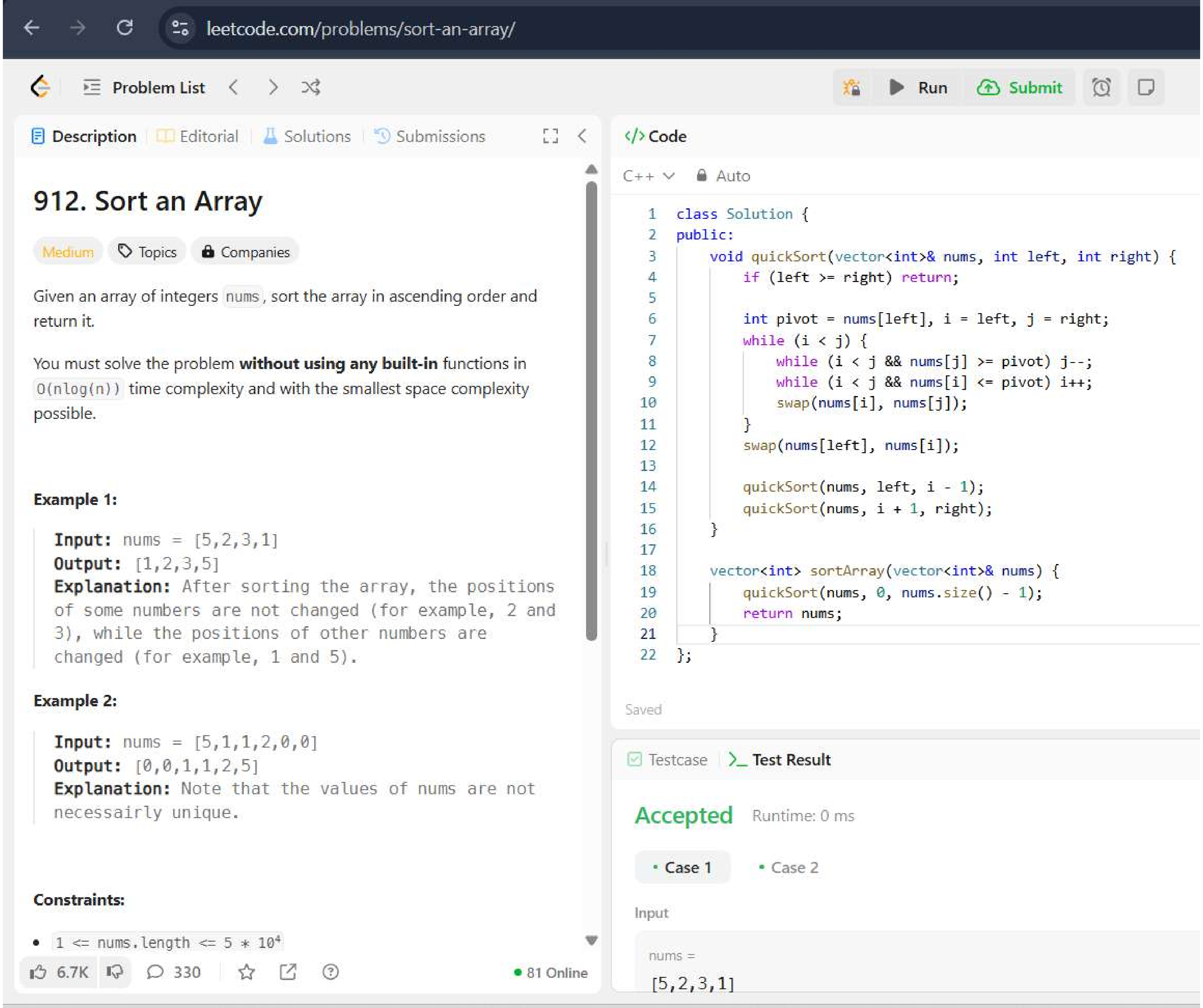
• Case 1 • Case 2 • Case 3

Input

a =

2

b =



912. Sort an Array

Given an array of integers `nums`, sort the array in ascending order and return it.

You must solve the problem **without using any built-in** functions in `O(nlog(n))` time complexity and with the smallest space complexity possible.

Example 1:

```
Input: nums = [5,2,3,1]
Output: [1,2,3,5]
Explanation: After sorting the array, the positions
of some numbers are not changed (for example, 2 and
3), while the positions of other numbers are
changed (for example, 1 and 5).
```

Example 2:

```
Input: nums = [5,1,1,2,0,0]
Output: [0,0,1,1,2,5]
Explanation: Note that the values of nums are not
necessairly unique.
```

Constraints:

- `1 <= nums.length <= 5 * 10`^{4}

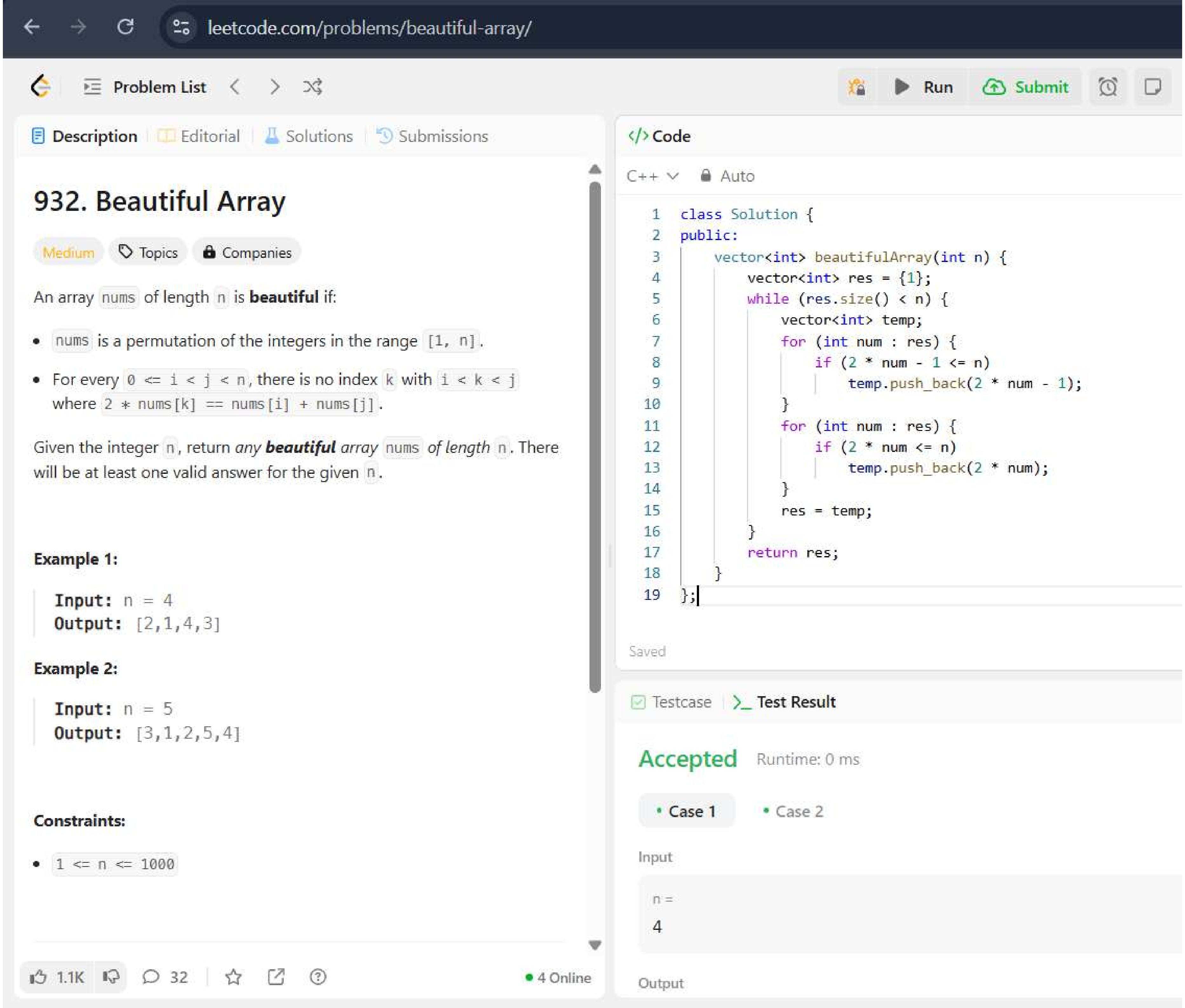
```cpp
class Solution {
public:
    void quickSort(vector<int>& nums, int left, int right) {
        if (left >= right) return;

        int pivot = nums[left], i = left, j = right;
        while (i < j) {
            while (i < j && nums[j] >= pivot) j--;
            while (i < j && nums[i] <= pivot) i++;
            swap(nums[i], nums[j]);
        }
        swap(nums[left], nums[i]);

        quickSort(nums, left, i - 1);
        quickSort(nums, i + 1, right);
    }

    vector<int> sortArray(vector<int>& nums) {
        quickSort(nums, 0, nums.size() - 1);
        return nums;
    }
};
```

Accepted Runtime: 0 ms

Input

nums =

[5,2,3,1]



932. Beautiful Array

An array `nums` of length `n` is **beautiful** if:

- `nums` is a permutation of the integers in the range `[1, n]`.
- For every `0 <= i < j < n`, there is no index `k` with `i < k < j` where `2 * nums[k] == nums[i] + nums[j]`.

Given the integer `n`, return *any **beautiful** array* `nums` *of length* `n`. There will be at least one valid answer for the given `n`.

Example 1:

```
Input: n = 4
Output: [2,1,4,3]
```

Example 2:

```
Input: n = 5
Output: [3,1,2,5,4]
```

Constraints:

- `1 <= n <= 1000`

```cpp
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> res = {1};
        while (res.size() < n) {
            vector<int> temp;
            for (int num : res) {
                if (2 * num - 1 <= n)
                    temp.push_back(2 * num - 1);
            }
            for (int num : res) {
                if (2 * num <= n)
                    temp.push_back(2 * num);
            }
            res = temp;
        }
        return res;
    }
};
```

Accepted Runtime: 0 ms

Input

n =

4

Output