SRI CHANDRASEKHARENDRA SARASWATHI VISWA MAHAVIDYALAYA

(UNIVERSITY ESTABLISHED UNDER SECTION 3 OF UGC ACT 1956)

ENATHUR, KANCHIPURAM – 631 561

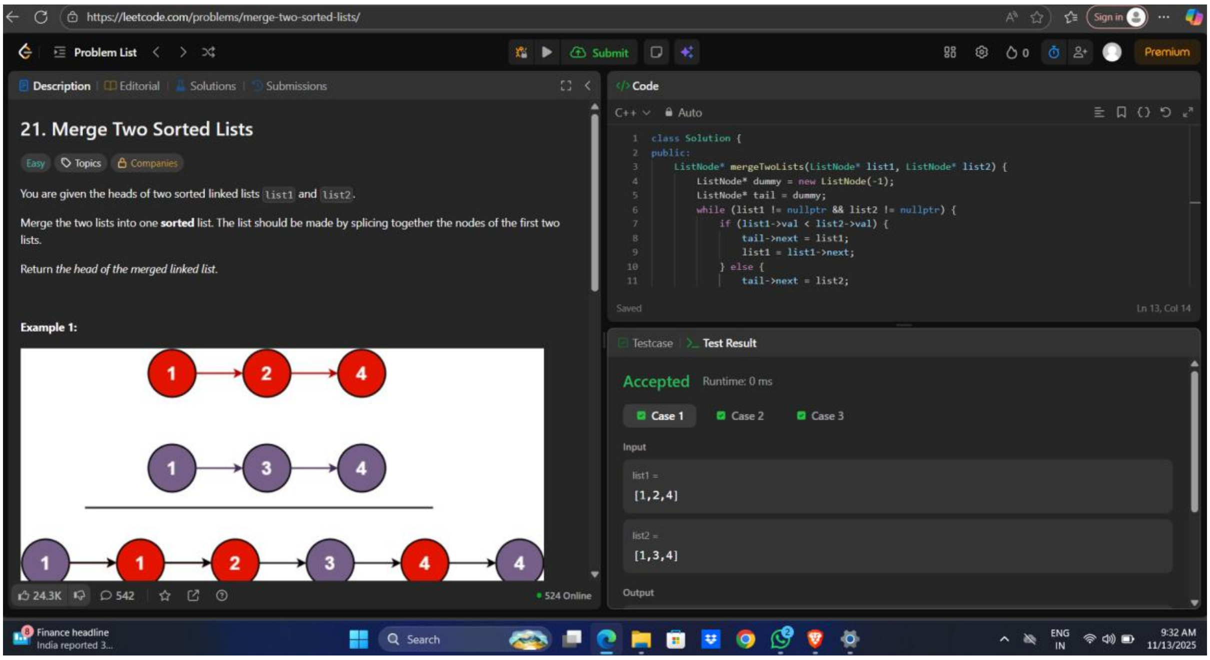
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Name: Ch. Manikanta Sai

Reg. No: 11249M007

Class: S7 II BTech(IT)

Course Name: OOPS Leet Code Problems



21. Merge Two Sorted Lists

Easy · Topics · Companies

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list.*

Example 1:

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        ListNode* dummy = new ListNode(-1);
        ListNode* tail = dummy;
        while (list1 != nullptr && list2 != nullptr) {
            if (list1->val < list2->val) {
                tail->next = list1;
                list1 = list1->next;
            } else {
                tail->next = list2;
```

Accepted Runtime: 0 ms

☑ Case 1 ☑ Case 2 ☑ Case 3

Input

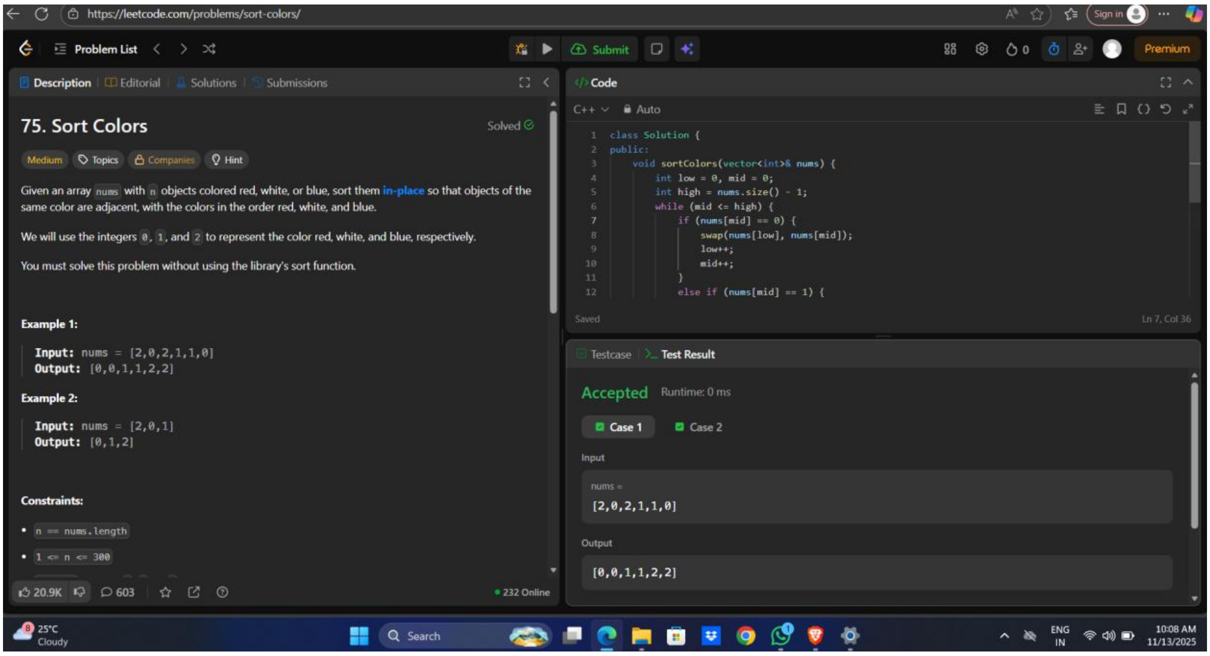
list1 =

```
[1,2,4]
```

list2 =

```
[1,3,4]
```

Output



75. Sort Colors

Solved ✓

Medium · Topics · Companies · Hint

Given an array `nums` with `n` objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers `0`, `1`, and `2` to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

```
Input: nums = [2,0,2,1,1,0]
Output: [0,0,1,1,2,2]
```

Example 2:

```
Input: nums = [2,0,1]
Output: [0,1,2]
```

Constraints:

- `n == nums.length`
- `1 <= n <= 300`

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low = 0, mid = 0;
        int high = nums.size() - 1;
        while (mid <= high) {
            if (nums[mid] == 0) {
                swap(nums[low], nums[mid]);
                low++;
                mid++;
            }
            else if (nums[mid] == 1) {
```

Accepted Runtime: 0 ms

☑ Case 1 ☑ Case 2

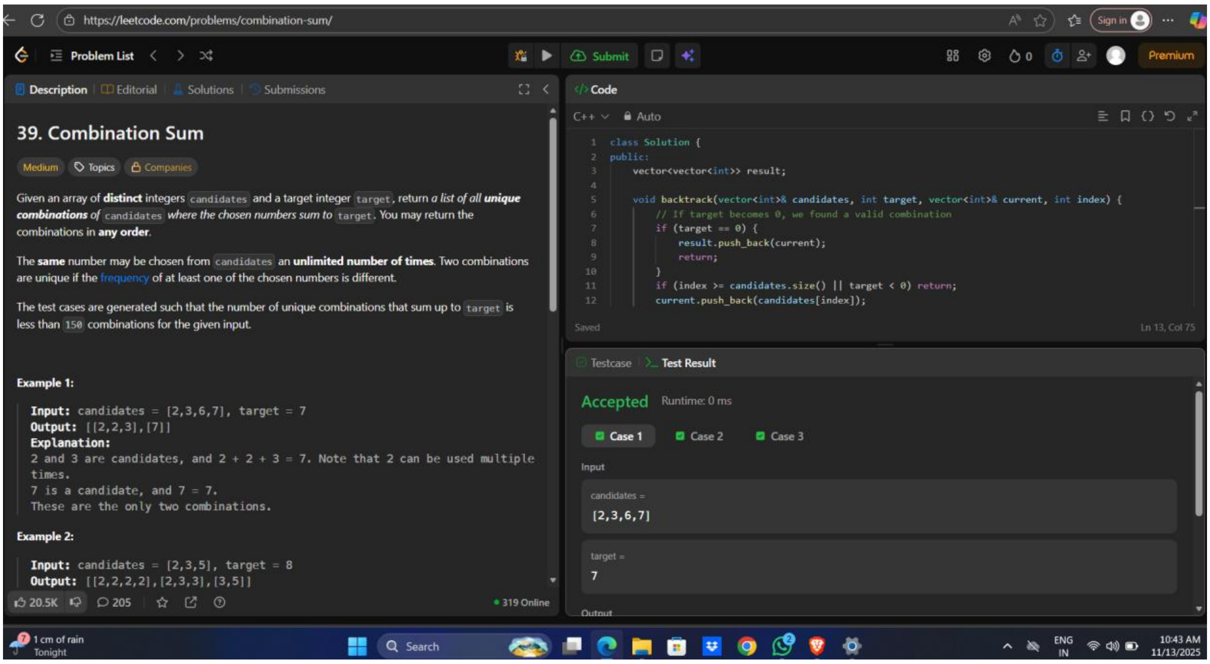
Input

nums =

```
[2,0,2,1,1,0]
```

Output

```
[0,0,1,1,2,2]
```



39. Combination Sum

Medium | Topics | Companies

Given an array of **distinct** integers `candidates` and a target integer `target`, return *a list of all **unique combinations** of* `candidates` *where the chosen numbers sum to* `target`*.* You may return the combinations in **any order**.

The **same** number may be chosen from `candidates` an **unlimited number of times**. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to `target` is less than `150` combinations for the given input.

Example 1:

```
Input: candidates = [2,3,6,7], target = 7
Output: [[2,2,3],[7]]
Explanation:
2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple
times.
7 is a candidate, and 7 = 7.
These are the only two combinations.
```

Example 2:

```
Input: candidates = [2,3,5], target = 8
Output: [[2,2,2,2],[2,3,3],[3,5]]
```

```cpp
class Solution {
public:
    vector<vector<int>> result;

    void backtrack(vector<int>& candidates, int target, vector<int>& current, int index) {
        // If target becomes 0, we found a valid combination
        if (target == 0) {
            result.push_back(current);
            return;
        }
        if (index >= candidates.size() || target < 0) return;
        current.push_back(candidates[index]);
```

Accepted Runtime: 0 ms

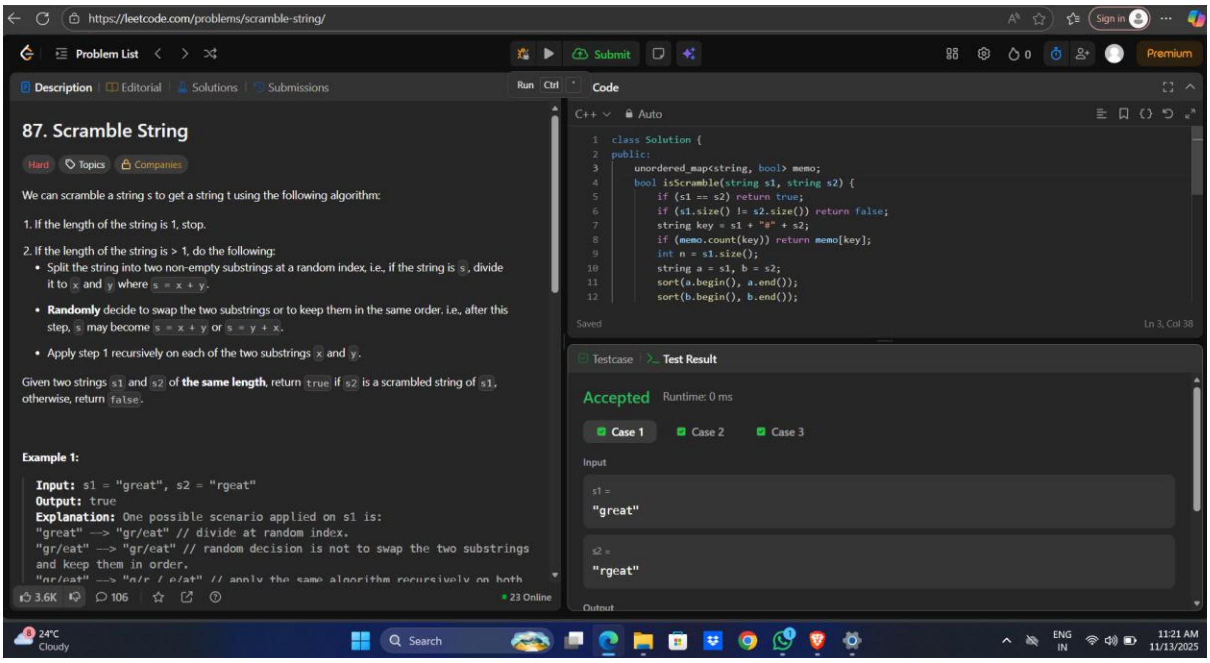
☑ Case 1 ☑ Case 2 ☑ Case 3

Input

candidates =
[2,3,6,7]

target =
7

Output



87. Scramble String

Hard | Topics | Companies

We can scramble a string s to get a string t using the following algorithm:

1. If the length of the string is 1, stop.
2. If the length of the string is > 1, do the following:
   - Split the string into two non-empty substrings at a random index, i.e., if the string is `s`, divide it to `x` and `y` where `s = x + y`.
   - **Randomly** decide to swap the two substrings or to keep them in the same order. i.e., after this step, `s` may become `s = x + y` or `s = y + x`.
   - Apply step 1 recursively on each of the two substrings `x` and `y`.

Given two strings `s1` and `s2` of **the same length**, return `true` if `s2` is a scrambled string of `s1`, otherwise, return `false`.

Example 1:

```
Input: s1 = "great", s2 = "rgeat"
Output: true
Explanation: One possible scenario applied on s1 is:
"great" —> "gr/eat" // divide at random index.
"gr/eat" —> "gr/eat" // random decision is not to swap the two substrings
and keep them in order.
"gr/eat" —> "g/r / e/at" // apply the same algorithm recursively on both
```

```cpp
class Solution {
public:
    unordered_map<string, bool> memo;
    bool isScramble(string s1, string s2) {
        if (s1 == s2) return true;
        if (s1.size() != s2.size()) return false;
        string key = s1 + "#" + s2;
        if (memo.count(key)) return memo[key];
        int n = s1.size();
        string a = s1, b = s2;
        sort(a.begin(), a.end());
        sort(b.begin(), b.end());
```

Accepted Runtime: 0 ms

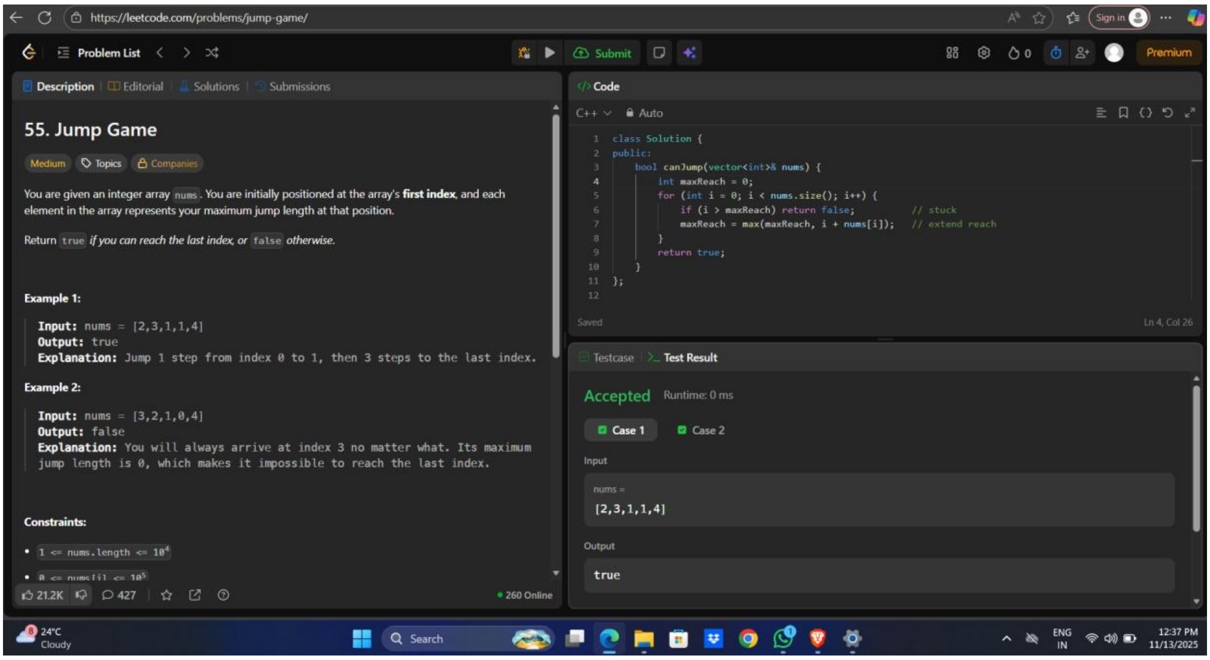
☑ Case 1 ☑ Case 2 ☑ Case 3

Input

s1 =
"great"

s2 =
"rgeat"

Output



55. Jump Game

Medium | Topics | Companies

You are given an integer array `nums`. You are initially positioned at the array's **first index**, and each element in the array represents your maximum jump length at that position.

Return `true` *if you can reach the last index, or* `false` *otherwise.*

Example 1:

```
Input: nums = [2,3,1,1,4]
Output: true
Explanation: Jump 1 step from index 0 to 1, then 3 steps to the last index.
```

Example 2:

```
Input: nums = [3,2,1,0,4]
Output: false
Explanation: You will always arrive at index 3 no matter what. Its maximum
jump length is 0, which makes it impossible to reach the last index.
```

Constraints:

- `1 <= nums.length <= 10^4`
- `0 <= nums[i] <= 10^5`

```cpp
class Solution {
public:
    bool canJump(vector<int>& nums) {
        int maxReach = 0;
        for (int i = 0; i < nums.size(); i++) {
            if (i > maxReach) return false;          // stuck
            maxReach = max(maxReach, i + nums[i]);   // extend reach
        }
        return true;
    }
};
```

Saved   Ln 4, Col 26

Accepted   Runtime: 0 ms

☑ Case 1   ☑ Case 2

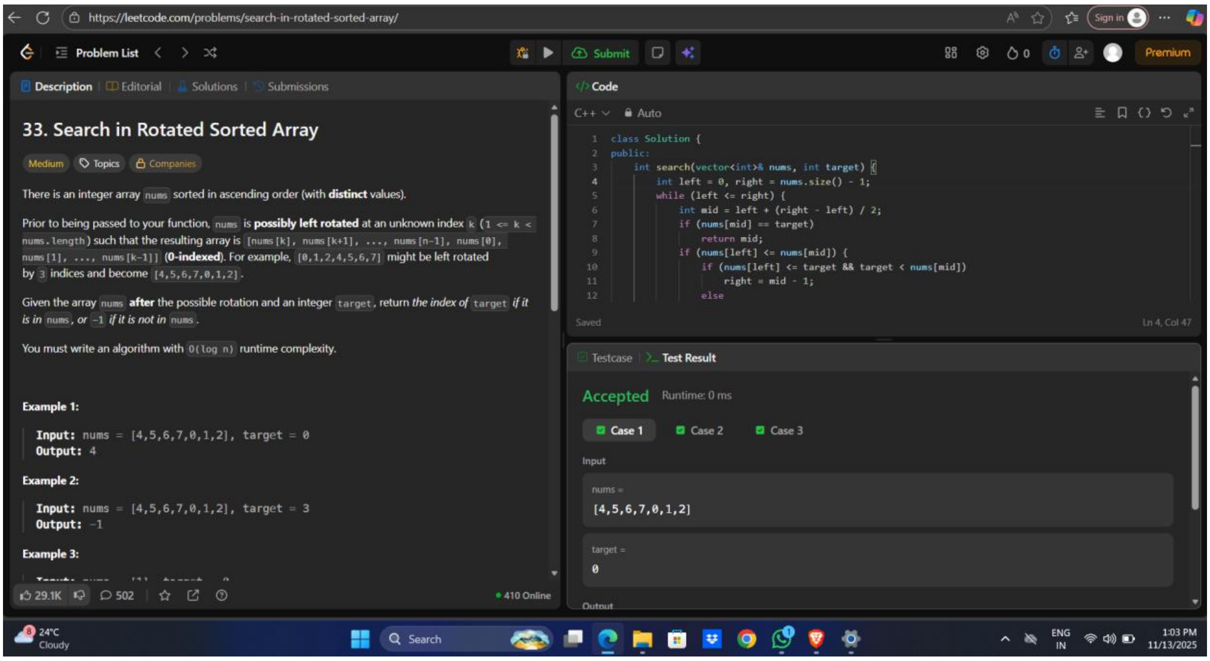
Input

nums =

```
[2,3,1,1,4]
```

Output

```
true
```



33. Search in Rotated Sorted Array

Medium | Topics | Companies

There is an integer array `nums` sorted in ascending order (with **distinct** values).

Prior to being passed to your function, `nums` is **possibly left rotated** at an unknown index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be left rotated by `3` indices and become `[4,5,6,7,0,1,2]`.

Given the array `nums` **after** the possible rotation and an integer `target`, return *the index of* `target` *if it is in* `nums`*, or* `-1` *if it is not in* `nums`.

You must write an algorithm with `O(log n)` runtime complexity.

Example 1:

```
Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

Example 3:

```
Input: nums = [1], target = 0
```

```cpp
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int left = 0, right = nums.size() - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target)
                return mid;
            if (nums[left] <= nums[mid]) {
                if (nums[left] <= target && target < nums[mid])
                    right = mid - 1;
                else
```

Saved   Ln 4, Col 47

Accepted   Runtime: 0 ms

☑ Case 1   ☑ Case 2   ☑ Case 3

Input

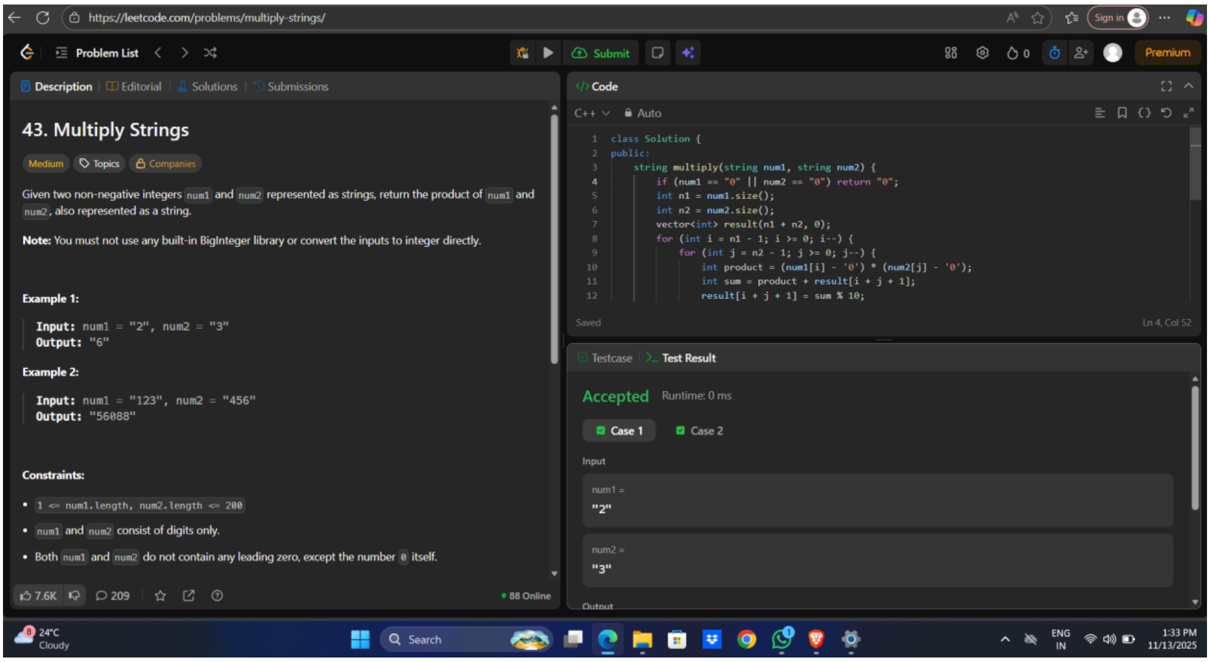
nums =

```
[4,5,6,7,0,1,2]
```

target =

```
0
```

Output



https://leetcode.com/problems/multiply-strings/

43. Multiply Strings

Given two non-negative integers `num1` and `num2` represented as strings, return the product of `num1` and `num2`, also represented as a string.

Note: You must not use any built-in BigInteger library or convert the inputs to integer directly.

Example 1:

```
Input: num1 = "2", num2 = "3"
Output: "6"
```

Example 2:

```
Input: num1 = "123", num2 = "456"
Output: "56088"
```

Constraints:

- `1 <= num1.length, num2.length <= 200`
- `num1` and `num2` consist of digits only.
- Both `num1` and `num2` do not contain any leading zero, except the number `0` itself.

```cpp
class Solution {
public:
    string multiply(string num1, string num2) {
        if (num1 == "0" || num2 == "0") return "0";
        int n1 = num1.size();
        int n2 = num2.size();
        vector<int> result(n1 + n2, 0);
        for (int i = n1 - 1; i >= 0; i--) {
            for (int j = n2 - 1; j >= 0; j--) {
                int product = (num1[i] - '0') * (num2[j] - '0');
                int sum = product + result[i + j + 1];
                result[i + j + 1] = sum % 10;
```

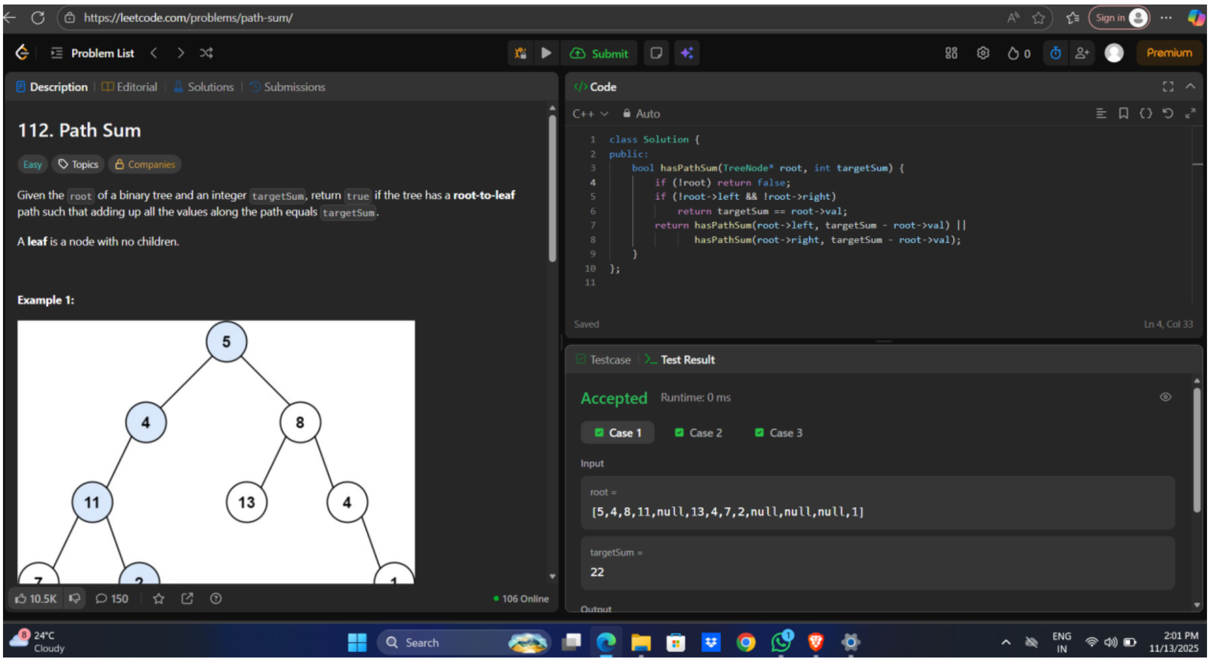
Accepted Runtime: 0 ms

Input

num1 =
"2"

num2 =
"3"

Output



https://leetcode.com/problems/path-sum/

112. Path Sum

Given the `root` of a binary tree and an integer `targetSum`, return `true` if the tree has a **root-to-leaf** path such that adding up all the values along the path equals `targetSum`.

A **leaf** is a node with no children.

Example 1:

```cpp
class Solution {
public:
    bool hasPathSum(TreeNode* root, int targetSum) {
        if (!root) return false;
        if (!root->left && !root->right)
            return targetSum == root->val;
        return hasPathSum(root->left, targetSum - root->val) ||
               hasPathSum(root->right, targetSum - root->val);
    }
};
```

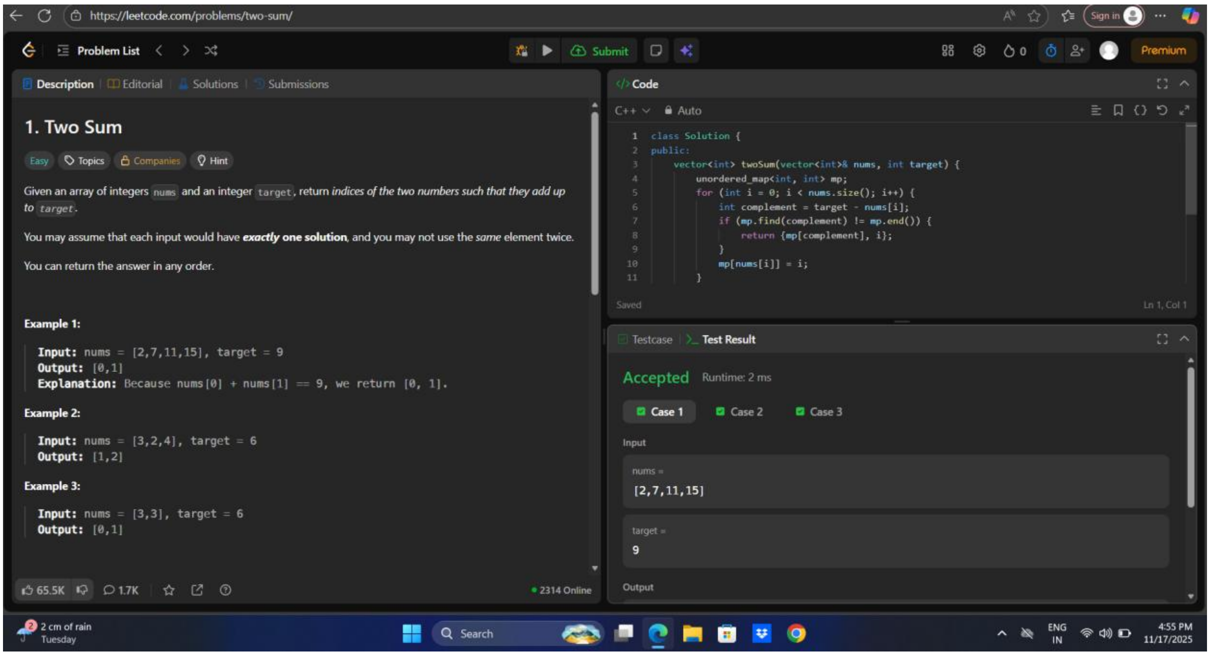
Accepted Runtime: 0 ms

Input

root =
[5,4,8,11,null,13,4,7,2,null,null,null,1]

targetSum =
22

Output



1. Two Sum

Easy · Topics · Companies · Hint

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Example 2:

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

Example 3:

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

```cpp
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> mp;
        for (int i = 0; i < nums.size(); i++) {
            int complement = target - nums[i];
            if (mp.find(complement) != mp.end()) {
                return {mp[complement], i};
            }
            mp[nums[i]] = i;
        }
```

Saved · Ln 1, Col 1

Accepted Runtime: 2 ms

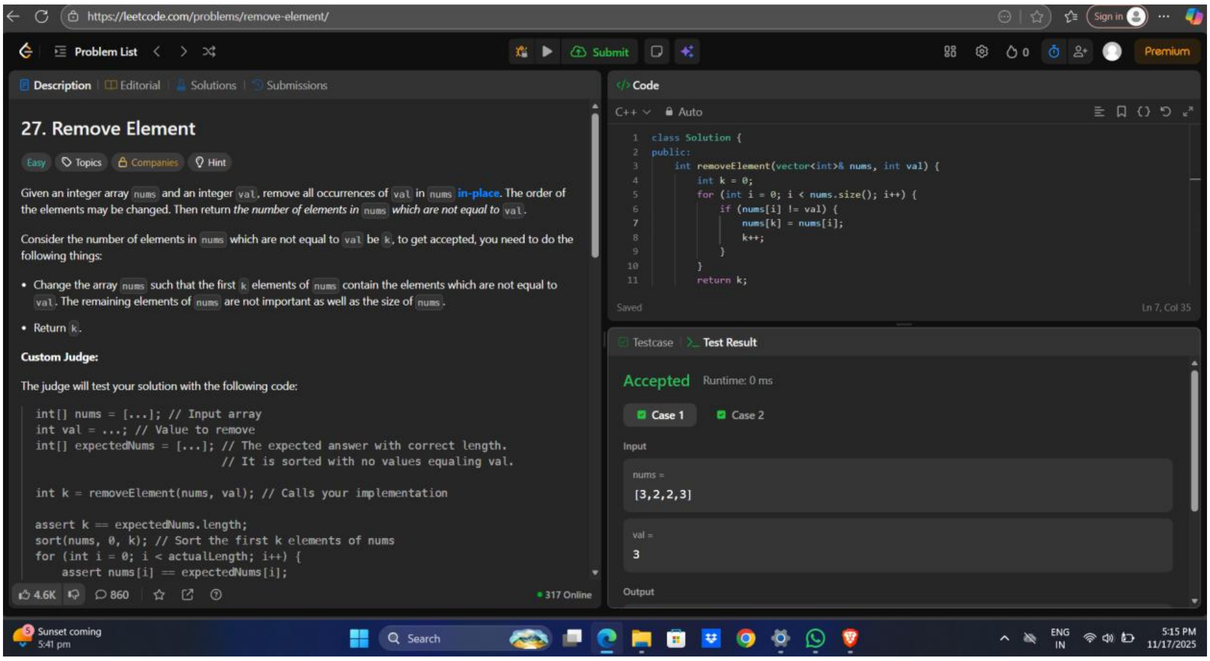
☑ Case 1 ☑ Case 2 ☑ Case 3

Input

```
nums =
[2,7,11,15]

target =
9
```

Output



27. Remove Element

Easy · Topics · Companies · Hint

Given an integer array `nums` and an integer `val`, remove all occurrences of `val` in `nums` in-place. The order of the elements may be changed. Then return *the number of elements in* `nums` *which are not equal to* `val`.

Consider the number of elements in `nums` which are not equal to `val` be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the elements which are not equal to `val`. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int val = ...; // Value to remove
int[] expectedNums = [...]; // The expected answer with correct length.
                            // It is sorted with no values equaling val.

int k = removeElement(nums, val); // Calls your implementation

assert k == expectedNums.length;
sort(nums, 0, k); // Sort the first k elements of nums
for (int i = 0; i < actualLength; i++) {
    assert nums[i] == expectedNums[i];
```

```cpp
class Solution {
public:
    int removeElement(vector<int>& nums, int val) {
        int k = 0;
        for (int i = 0; i < nums.size(); i++) {
            if (nums[i] != val) {
                nums[k] = nums[i];
                k++;
            }
        }
        return k;
```

Saved · Ln 7, Col 35

Accepted Runtime: 0 ms

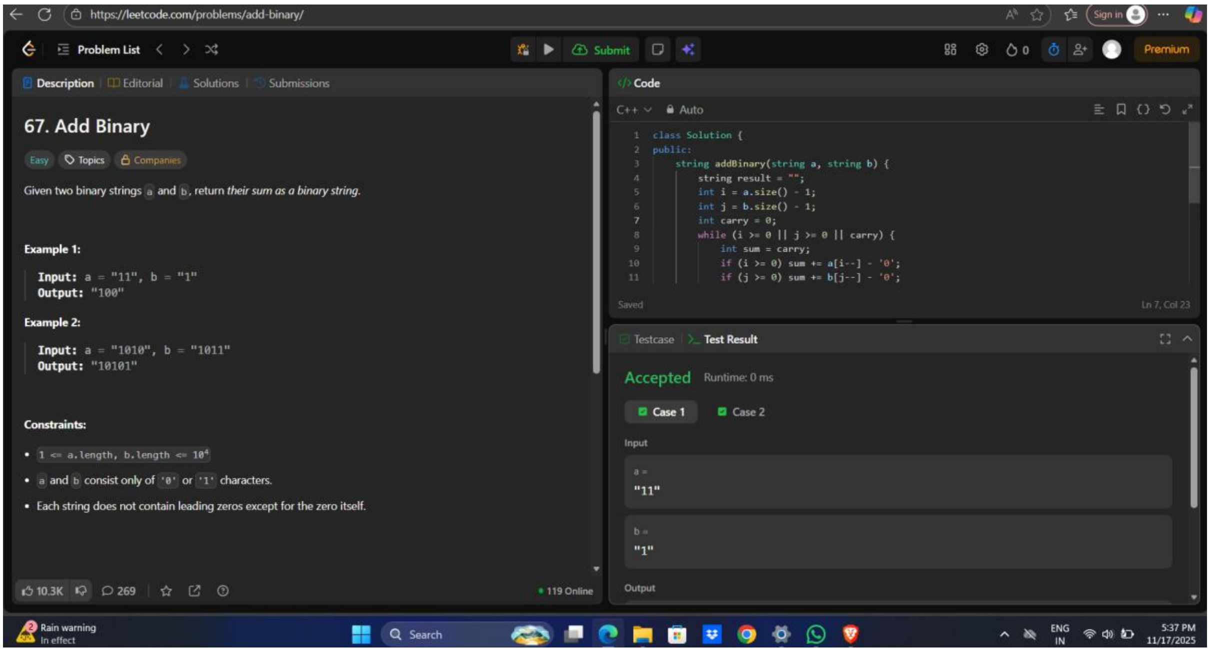
☑ Case 1 ☑ Case 2

Input

```
nums =
[3,2,2,3]

val =
3
```

Output



67. Add Binary

Given two binary strings `a` and `b`, return *their sum as a binary string*.

Example 1:

```
Input: a = "11", b = "1"
Output: "100"
```

Example 2:

```
Input: a = "1010", b = "1011"
Output: "10101"
```

Constraints:

- `1 <= a.length, b.length <= 10⁴`
- `a` and `b` consist only of `'0'` or `'1'` characters.
- Each string does not contain leading zeros except for the zero itself.

Code

C++ Auto

```cpp
class Solution {
public:
    string addBinary(string a, string b) {
        string result = "";
        int i = a.size() - 1;
        int j = b.size() - 1;
        int carry = 0;
        while (i >= 0 || j >= 0 || carry) {
            int sum = carry;
            if (i >= 0) sum += a[i--] - '0';
            if (j >= 0) sum += b[j--] - '0';
```

Saved Ln 7, Col 23

Testcase | Test Result

Accepted Runtime: 0 ms

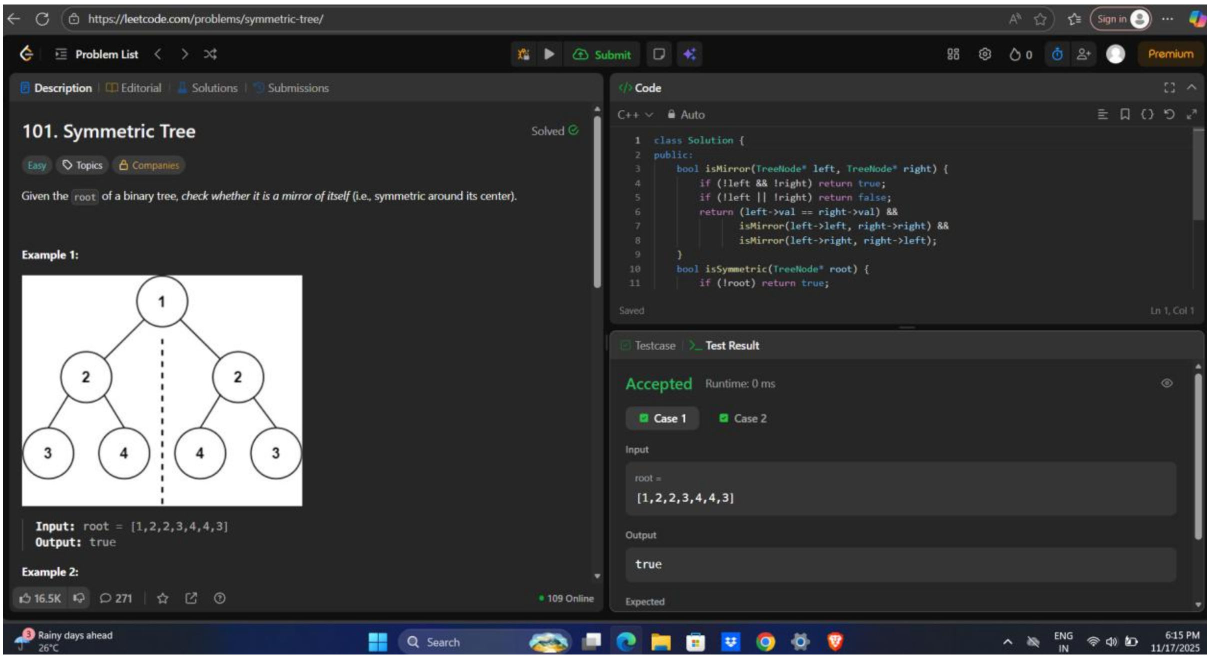
Case 1 Case 2

Input

a =
"11"

b =
"1"

Output



101. Symmetric Tree

Given the `root` of a binary tree, *check whether it is a mirror of itself* (i.e., symmetric around its center).

Example 1:

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

Example 2:

Code

C++ Auto

```cpp
class Solution {
public:
    bool isMirror(TreeNode* left, TreeNode* right) {
        if (!left && !right) return true;
        if (!left || !right) return false;
        return (left->val == right->val) &&
               isMirror(left->left, right->right) &&
               isMirror(left->right, right->left);
    }
    bool isSymmetric(TreeNode* root) {
        if (!root) return true;
```

Saved Ln 1, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

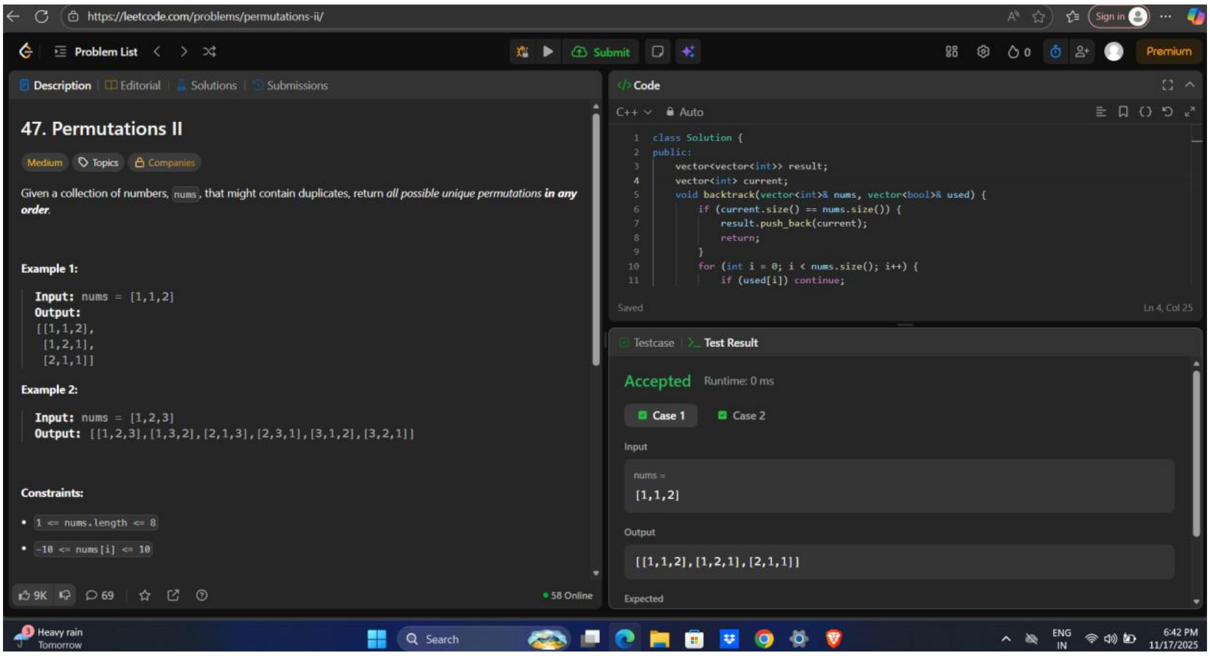
Input

root =
[1,2,2,3,4,4,3]

Output

true

Expected



47. Permutations II

Given a collection of numbers, `nums`, that might contain duplicates, return *all possible unique permutations* ***in any order***.

Example 1:

```
Input: nums = [1,1,2]
Output:
[[1,1,2],
 [1,2,1],
 [2,1,1]]
```

Example 2:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Constraints:

- `1 <= nums.length <= 8`
- `-10 <= nums[i] <= 10`

```cpp
class Solution {
public:
    vector<vector<int>> result;
    vector<int> current;
    void backtrack(vector<int>& nums, vector<bool>& used) {
        if (current.size() == nums.size()) {
            result.push_back(current);
            return;
        }
        for (int i = 0; i < nums.size(); i++) {
            if (used[i]) continue;
```

Accepted Runtime: 0 ms

Input

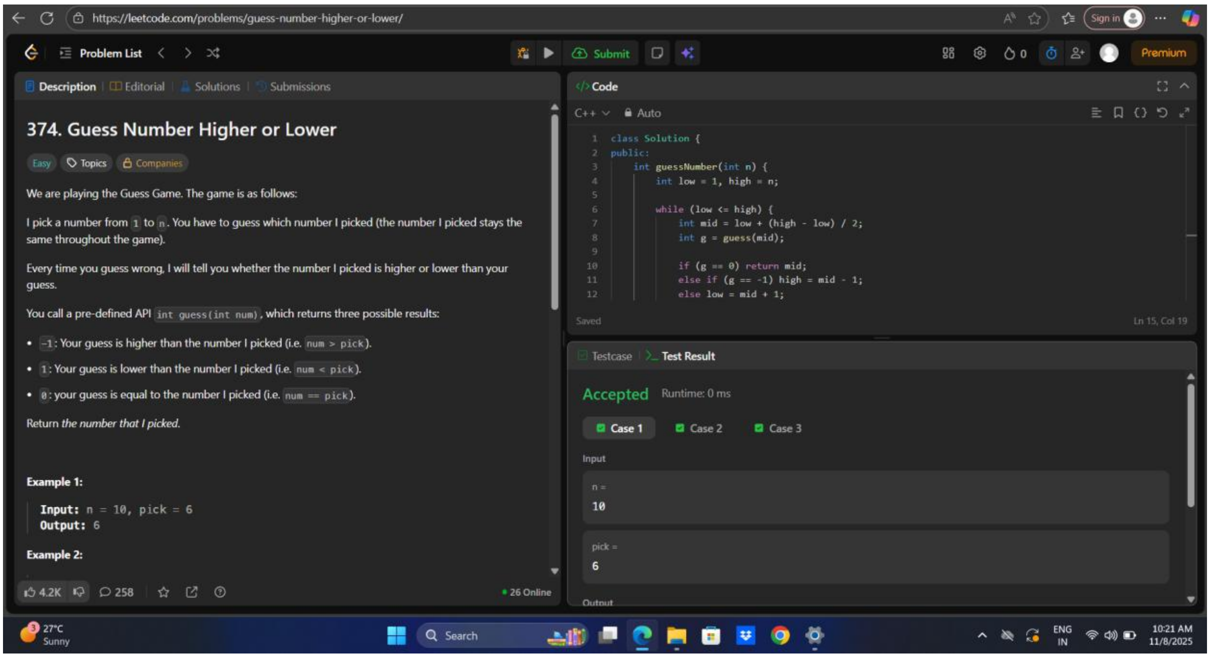
nums =

```
[1,1,2]
```

Output

```
[[1,1,2],[1,2,1],[2,1,1]]
```

Expected



374. Guess Number Higher or Lower

We are playing the Guess Game. The game is as follows:

I pick a number from `1` to `n`. You have to guess which number I picked (the number I picked stays the same throughout the game).

Every time you guess wrong, I will tell you whether the number I picked is higher or lower than your guess.

You call a pre-defined API `int guess(int num)`, which returns three possible results:

- `-1`: Your guess is higher than the number I picked (i.e. `num > pick`).
- `1`: Your guess is lower than the number I picked (i.e. `num < pick`).
- `0`: your guess is equal to the number I picked (i.e. `num == pick`).

Return *the number that I picked*.

Example 1:

```
Input: n = 10, pick = 6
Output: 6
```

Example 2:

```cpp
class Solution {
public:
    int guessNumber(int n) {
        int low = 1, high = n;

        while (low <= high) {
            int mid = low + (high - low) / 2;
            int g = guess(mid);

            if (g == 0) return mid;
            else if (g == -1) high = mid - 1;
            else low = mid + 1;
```

Accepted Runtime: 0 ms

Input

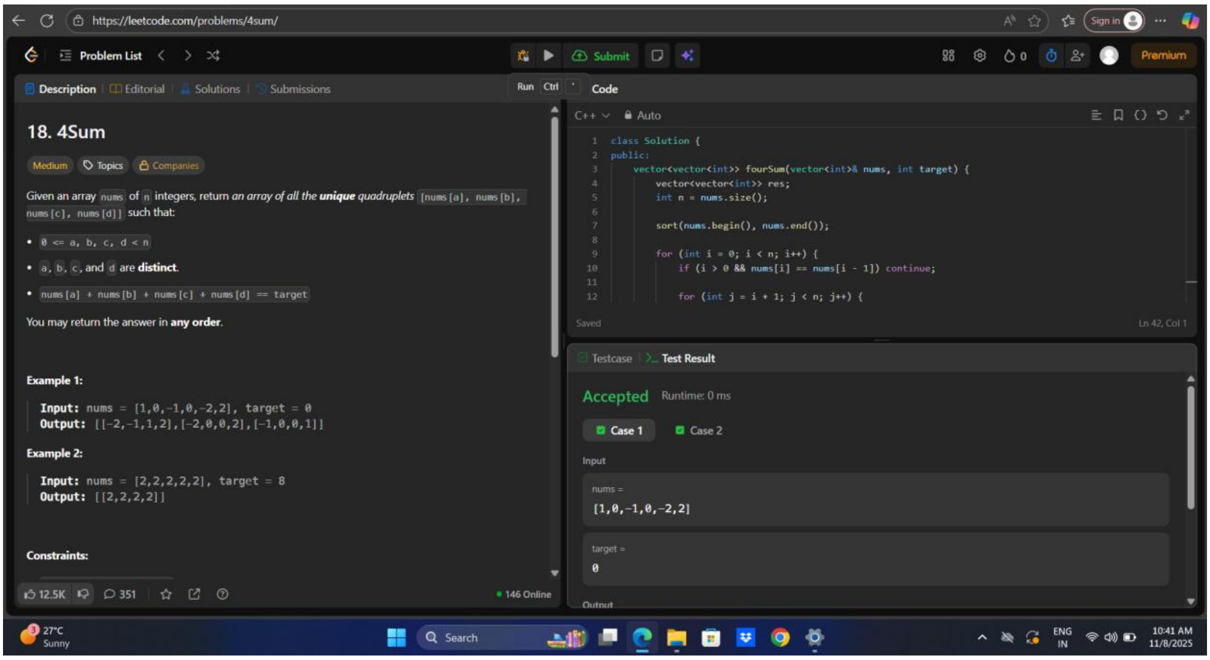
n =

```
10
```

pick =

```
6
```

Output



18. 4Sum

Medium Topics Companies

Given an array `nums` of `n` integers, return *an array of all the **unique** quadruplets* `[nums[a], nums[b], nums[c], nums[d]]` such that:

- `0 <= a, b, c, d < n`
- `a`, `b`, `c`, and `d` are **distinct**.
- `nums[a] + nums[b] + nums[c] + nums[d] == target`

You may return the answer in **any order**.

Example 1:

```
Input: nums = [1,0,-1,0,-2,2], target = 0
Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]
```

Example 2:

```
Input: nums = [2,2,2,2,2], target = 8
Output: [[2,2,2,2]]
```

Constraints:

```cpp
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        vector<vector<int>> res;
        int n = nums.size();

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

        for (int i = 0; i < n; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) continue;

            for (int j = i + 1; j < n; j++) {
```

Saved Ln 42, Col 1

Accepted Runtime: 0 ms

Case 1 Case 2

Input

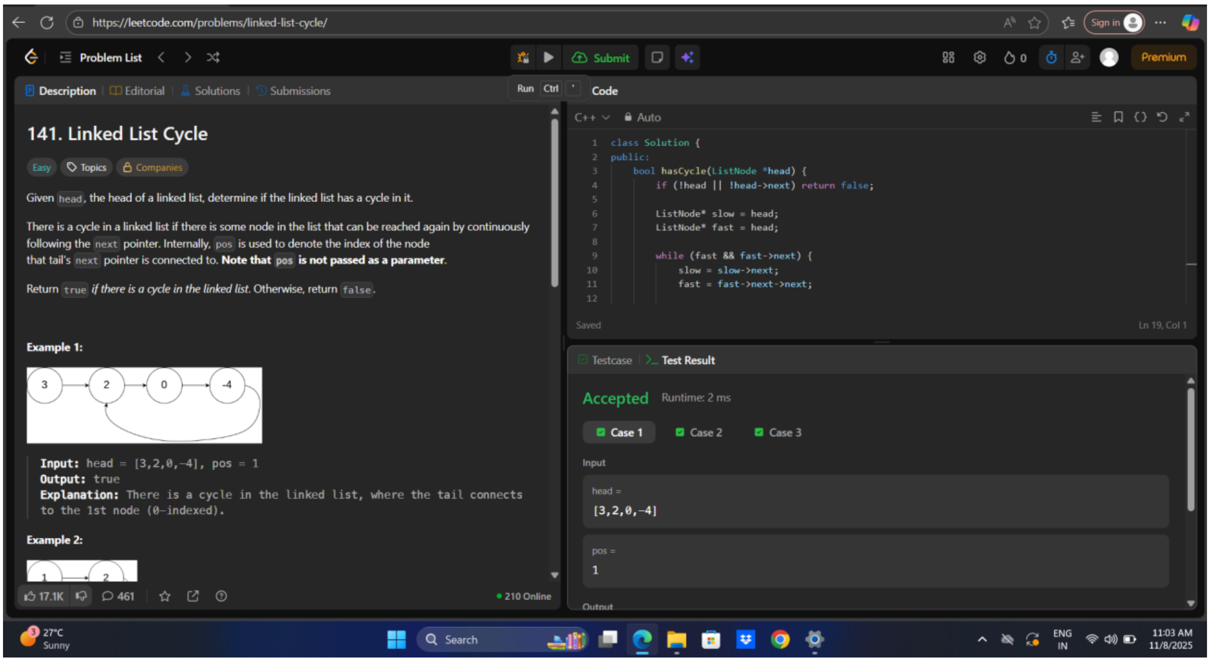
nums =

```
[1,0,-1,0,-2,2]
```

target =

```
0
```

Output



141. Linked List Cycle

Easy Topics Companies

Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to. **Note that** `pos` **is not passed as a parameter**.

Return `true` *if there is a cycle in the linked list*. Otherwise, return `false`.

Example 1:

```
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects
to the 1st node (0-indexed).
```

Example 2:

```cpp
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (!head || !head->next) return false;

        ListNode* slow = head;
        ListNode* fast = head;

        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

```

Saved Ln 19, Col 1

Accepted Runtime: 2 ms

Case 1 Case 2 Case 3

Input

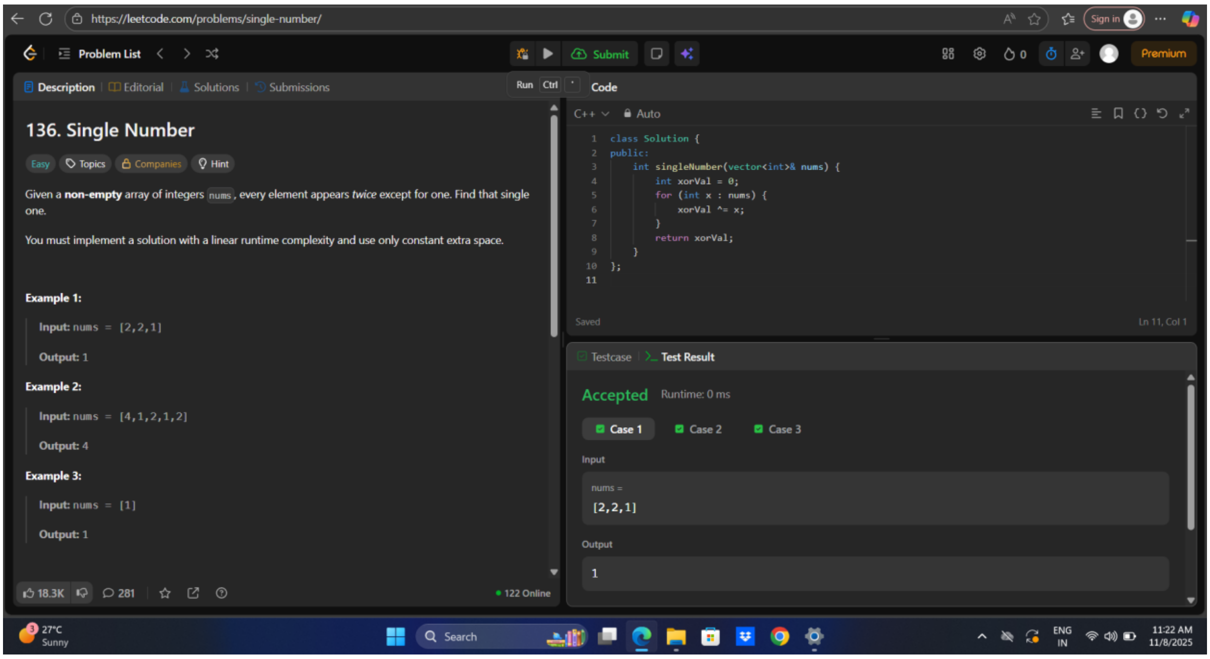
head =

```
[3,2,0,-4]
```

pos =

```
1
```

Output



136. Single Number

Easy Topics Companies Hint

Given a **non-empty** array of integers `nums`, every element appears *twice* except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:

Input: nums = [2,2,1]

Output: 1

Example 2:

Input: nums = [4,1,2,1,2]

Output: 4

Example 3:

Input: nums = [1]

Output: 1

```cpp
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        int xorVal = 0;
        for (int x : nums) {
            xorVal ^= x;
        }
        return xorVal;
    }
};
```

Accepted Runtime: 0 ms

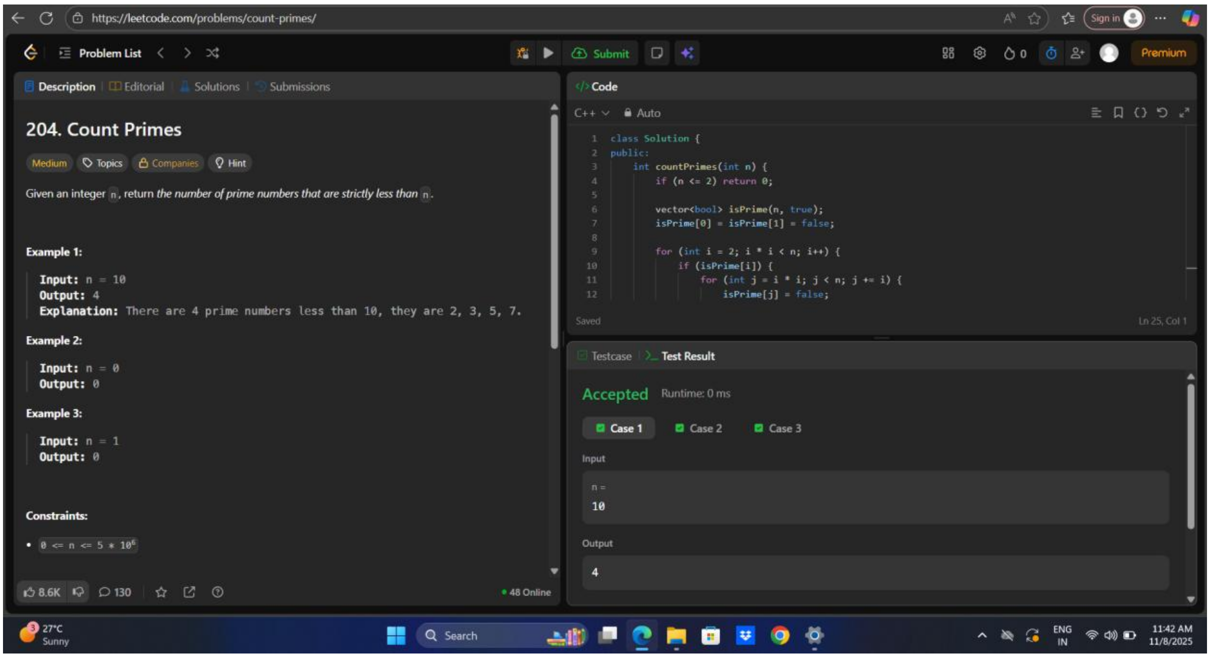
☑ Case 1 ☑ Case 2 ☑ Case 3

Input

```
nums =
[2,2,1]
```

Output

```
1
```



204. Count Primes

Medium Topics Companies Hint

Given an integer `n`, return *the number of prime numbers that are strictly less than* `n`.

Example 1:

```
Input: n = 10
Output: 4
Explanation: There are 4 prime numbers less than 10, they are 2, 3, 5, 7.
```

Example 2:

```
Input: n = 0
Output: 0
```

Example 3:

```
Input: n = 1
Output: 0
```

Constraints:

- $0 <= n <= 5 * 10^6$

```cpp
class Solution {
public:
    int countPrimes(int n) {
        if (n <= 2) return 0;

        vector<bool> isPrime(n, true);
        isPrime[0] = isPrime[1] = false;

        for (int i = 2; i * i < n; i++) {
            if (isPrime[i]) {
                for (int j = i * i; j < n; j += i) {
                    isPrime[j] = false;
```

Accepted Runtime: 0 ms

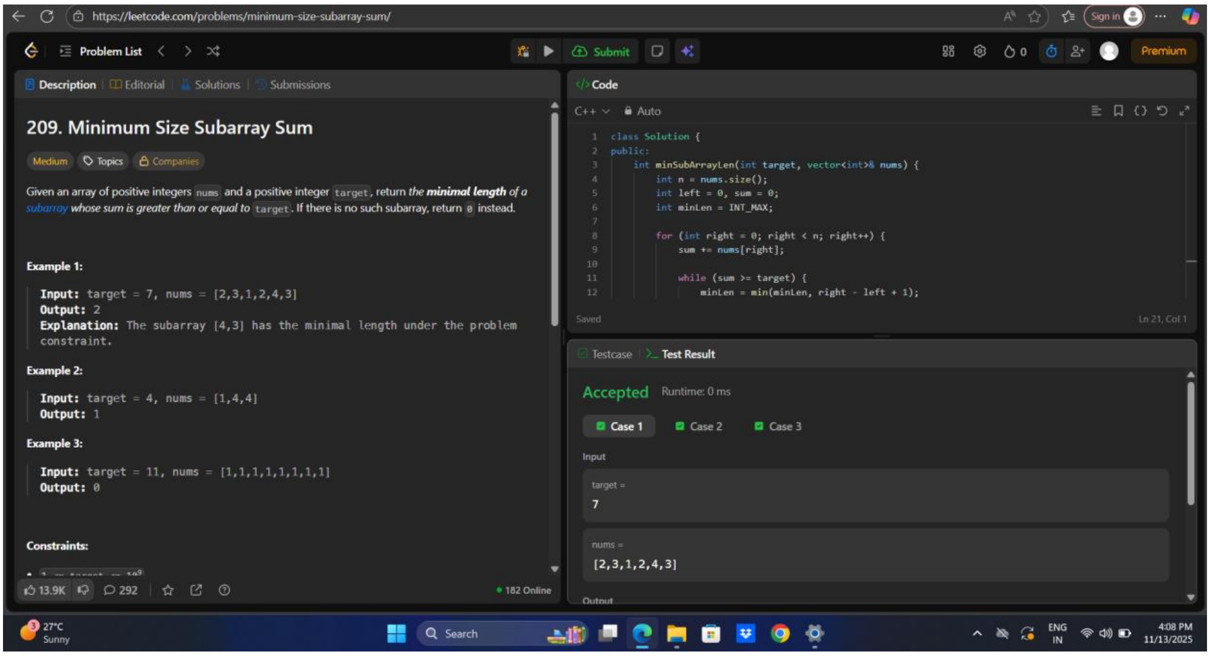
☑ Case 1 ☑ Case 2 ☑ Case 3

Input

```
n =
10
```

Output

```
4
```



209. Minimum Size Subarray Sum

Medium · Topics · Companies

Given an array of positive integers `nums` and a positive integer `target`, return *the **minimal length** of a subarray whose sum is greater than or equal to* `target`. If there is no such subarray, return `0` instead.

Example 1:

```
Input: target = 7, nums = [2,3,1,2,4,3]
Output: 2
Explanation: The subarray [4,3] has the minimal length under the problem
constraint.
```

Example 2:

```
Input: target = 4, nums = [1,4,4]
Output: 1
```

Example 3:

```
Input: target = 11, nums = [1,1,1,1,1,1,1,1]
Output: 0
```

Constraints:

```cpp
class Solution {
public:
    int minSubArrayLen(int target, vector<int>& nums) {
        int n = nums.size();
        int left = 0, sum = 0;
        int minLen = INT_MAX;

        for (int right = 0; right < n; right++) {
            sum += nums[right];

            while (sum >= target) {
                minLen = min(minLen, right - left + 1);
```

Saved · Ln 21, Col 1

Accepted Runtime: 0 ms

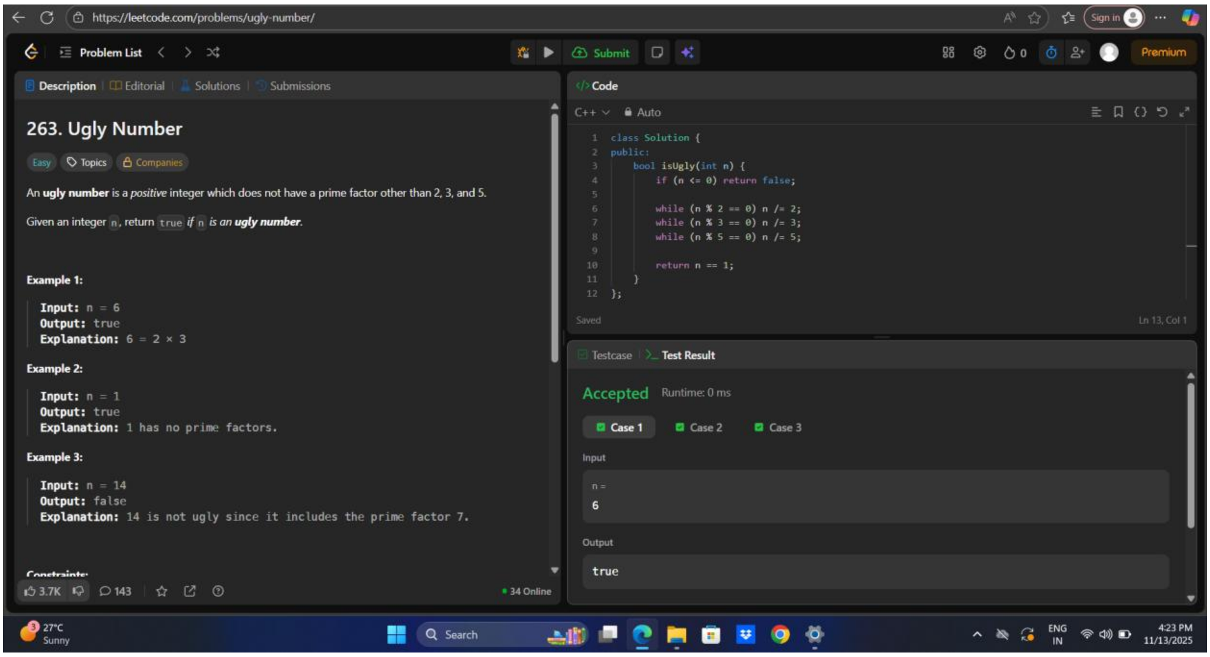
Case 1 · Case 2 · Case 3

Input

target =
7

nums =
[2,3,1,2,4,3]

Output



263. Ugly Number

Easy · Topics · Companies

An **ugly number** is a *positive* integer which does not have a prime factor other than 2, 3, and 5.

Given an integer `n`, return `true` *if* `n` *is an **ugly number***.

Example 1:

```
Input: n = 6
Output: true
Explanation: 6 = 2 × 3
```

Example 2:

```
Input: n = 1
Output: true
Explanation: 1 has no prime factors.
```

Example 3:

```
Input: n = 14
Output: false
Explanation: 14 is not ugly since it includes the prime factor 7.
```

Constraints:

```cpp
class Solution {
public:
    bool isUgly(int n) {
        if (n <= 0) return false;

        while (n % 2 == 0) n /= 2;
        while (n % 3 == 0) n /= 3;
        while (n % 5 == 0) n /= 5;

        return n == 1;
    }
};
```

Saved · Ln 13, Col 1

Accepted Runtime: 0 ms

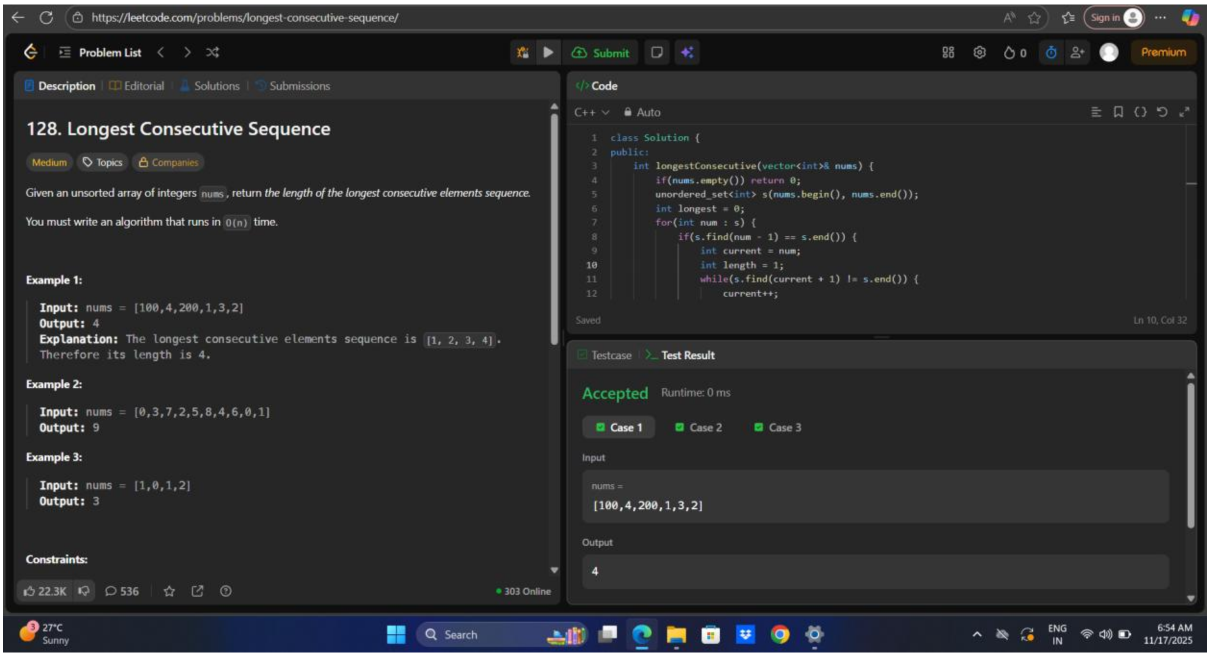
Case 1 · Case 2 · Case 3

Input

n =
6

Output

true


https://leetcode.com/problems/longest-consecutive-sequence/
Problem List
Submit
Premium
Description | Editorial | Solutions | Submissions
Code
Testcase | Test Result
22.3K
536
303 Online
27°C
Sunny
Search
ENG
IN
6:54 AM
11/17/2025

128. Longest Consecutive Sequence

Medium · Topics · Companies

Given an unsorted array of integers `nums`, return *the length of the longest consecutive elements sequence.*

You must write an algorithm that runs in `O(n)` time.

Example 1:

```
Input: nums = [100,4,200,1,3,2]
Output: 4
Explanation: The longest consecutive elements sequence is [1, 2, 3, 4].
Therefore its length is 4.
```

Example 2:

```
Input: nums = [0,3,7,2,5,8,4,6,0,1]
Output: 9
```

Example 3:

```
Input: nums = [1,0,1,2]
Output: 3
```

Constraints:

C++ · Auto

```cpp
class Solution {
public:
    int longestConsecutive(vector<int>& nums) {
        if(nums.empty()) return 0;
        unordered_set<int> s(nums.begin(), nums.end());
        int longest = 0;
        for(int num : s) {
            if(s.find(num - 1) == s.end()) {
                int current = num;
                int length = 1;
                while(s.find(current + 1) != s.end()) {
                    current++;
```

Saved · Ln 10, Col 32

Accepted Runtime: 0 ms

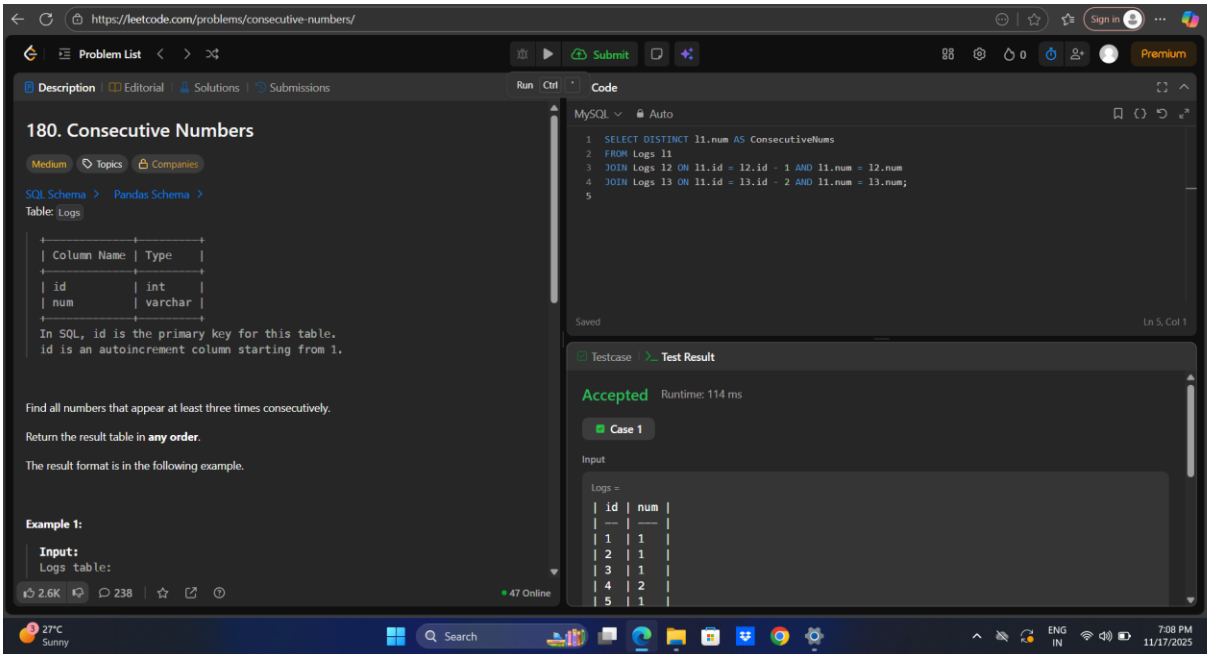
Case 1 · Case 2 · Case 3

Input

```
nums =
[100,4,200,1,3,2]
```

Output

```
4
```



180. Consecutive Numbers

Medium · Topics · Companies

SQL Schema > Pandas Schema >

Table: `Logs`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| num         | varchar |
+-------------+---------+
In SQL, id is the primary key for this table.
id is an autoincrement column starting from 1.
```

Find all numbers that appear at least three times consecutively.

Return the result table in **any order**.

The result format is in the following example.

Example 1:

```
Input:
Logs table:
```

MySQL · Auto

```sql
SELECT DISTINCT l1.num AS ConsecutiveNums
FROM Logs l1
JOIN Logs l2 ON l1.id = l2.id - 1 AND l1.num = l2.num
JOIN Logs l3 ON l1.id = l3.id - 2 AND l1.num = l3.num;
```

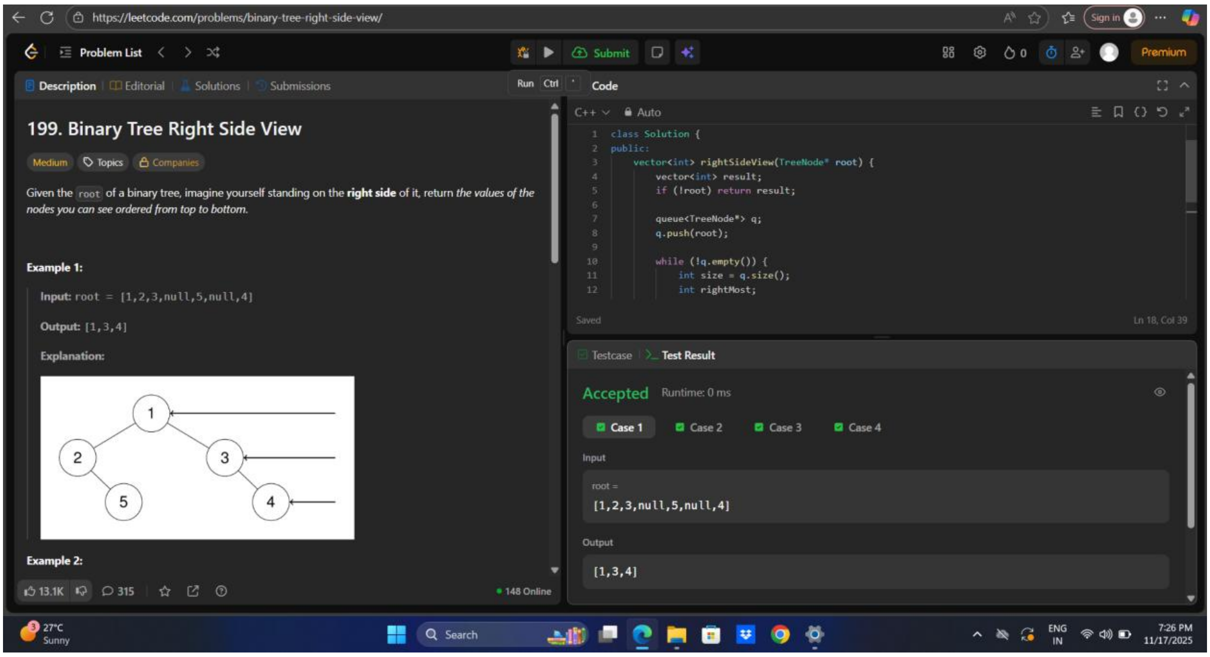
Saved · Ln 5, Col 1

Accepted Runtime: 114 ms

Case 1

Input

```
Logs =
| id | num |
| -- | --- |
| 1  | 1   |
| 2  | 1   |
| 3  | 1   |
| 4  | 2   |
| 5  | 1   |
```



https://leetcode.com/problems/binary-tree-right-side-view/

Problem List

Submit

Premium

Description | Editorial | Solutions | Submissions

199. Binary Tree Right Side View

Medium | Topics | Companies

Given the `root` of a binary tree, imagine yourself standing on the **right side** of it, return *the values of the nodes you can see ordered from top to bottom.*

Example 1:

Input: `root = [1,2,3,null,5,null,4]`

Output: `[1,3,4]`

Explanation:

Example 2:

13.1K | 315 | 148 Online

Code

C++ | Auto

```cpp
class Solution {
public:
    vector<int> rightSideView(TreeNode* root) {
        vector<int> result;
        if (!root) return result;

        queue<TreeNode*> q;
        q.push(root);

        while (!q.empty()) {
            int size = q.size();
            int rightMost;
```

Saved | Ln 18, Col 39

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3 | Case 4

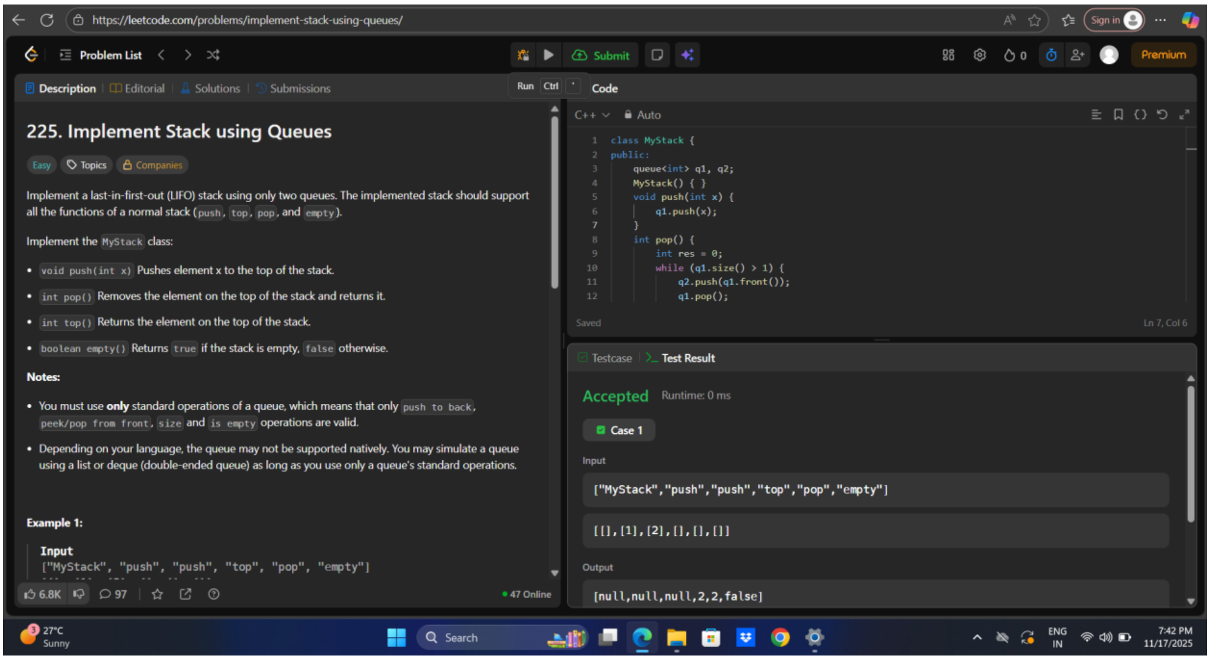
Input

```
root =
[1,2,3,null,5,null,4]
```

Output

```
[1,3,4]
```

27°C Sunny | Search | ENG IN | 7:26 PM 11/17/2025



https://leetcode.com/problems/implement-stack-using-queues/

Problem List

Submit

Premium

Description | Editorial | Solutions | Submissions

225. Implement Stack using Queues

Easy | Topics | Companies

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (`push`, `top`, `pop`, and `empty`).

Implement the `MyStack` class:

- `void push(int x)` Pushes element x to the top of the stack.
- `int pop()` Removes the element on the top of the stack and returns it.
- `int top()` Returns the element on the top of the stack.
- `boolean empty()` Returns `true` if the stack is empty, `false` otherwise.

Notes:

- You must use **only** standard operations of a queue, which means that only `push to back`, `peek/pop from front`, `size` and `is empty` operations are valid.
- Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deque (double-ended queue) as long as you use only a queue's standard operations.

Example 1:

```
Input
["MyStack", "push", "push", "top", "pop", "empty"]
```

6.8K | 97 | 47 Online

Code

C++ | Auto

```cpp
class MyStack {
public:
    queue<int> q1, q2;
    MyStack() { }
    void push(int x) {
        q1.push(x);
    }
    int pop() {
        int res = 0;
        while (q1.size() > 1) {
            q2.push(q1.front());
            q1.pop();
```

Saved | Ln 7, Col 6

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1

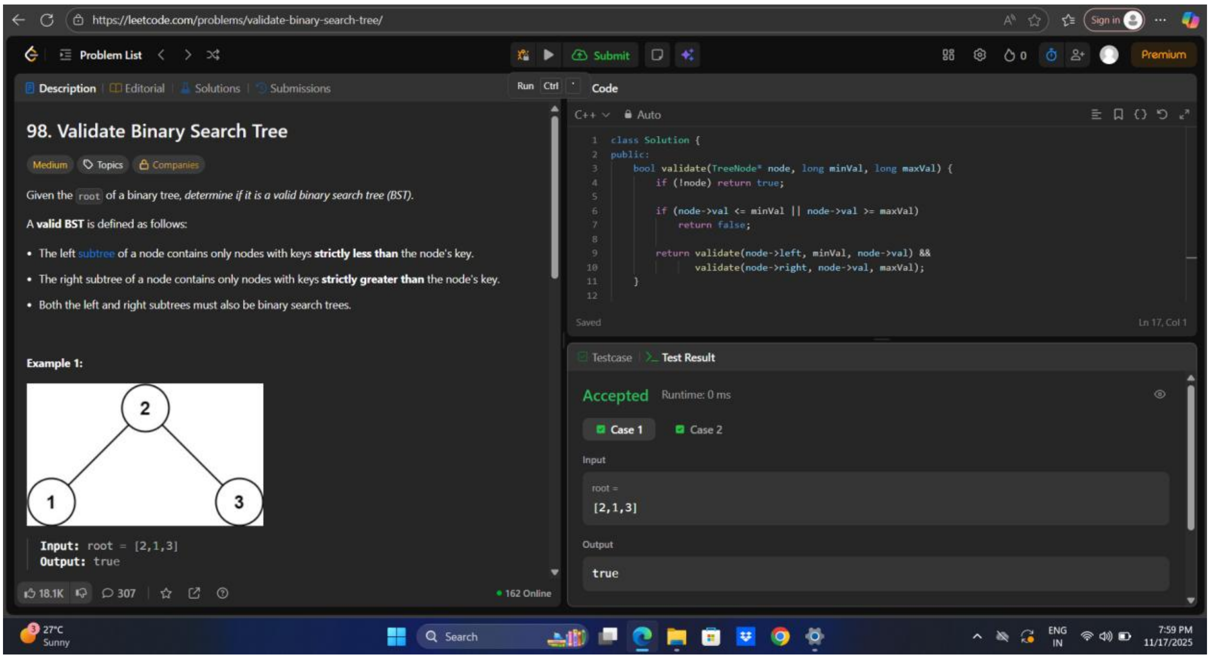
Input

```
["MyStack","push","push","top","pop","empty"]
[[],[1],[2],[],[],[]]
```

Output

```
[null,null,null,2,2,false]
```

27°C Sunny | Search | ENG IN | 7:42 PM 11/17/2025



98. Validate Binary Search Tree

Given the `root` of a binary tree, *determine if it is a valid binary search tree (BST)*.

A **valid BST** is defined as follows:

- The left subtree of a node contains only nodes with keys **strictly less than** the node's key.
- The right subtree of a node contains only nodes with keys **strictly greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

Example 1:

```
Input: root = [2,1,3]
Output: true
```

```cpp
class Solution {
public:
    bool validate(TreeNode* node, long minVal, long maxVal) {
        if (!node) return true;

        if (node->val <= minVal || node->val >= maxVal)
            return false;

        return validate(node->left, minVal, node->val) &&
               validate(node->right, node->val, maxVal);
    }

```

Accepted Runtime: 0 ms

Input

root =
[2,1,3]

Output

true