DESIGN AND ANALYSIS OF ALGORITHMS
HOLIDAY ASSIGNMENT

Name: N. Manasa Priya
Roll No: 2211CS020366
Section: AIML-THETA

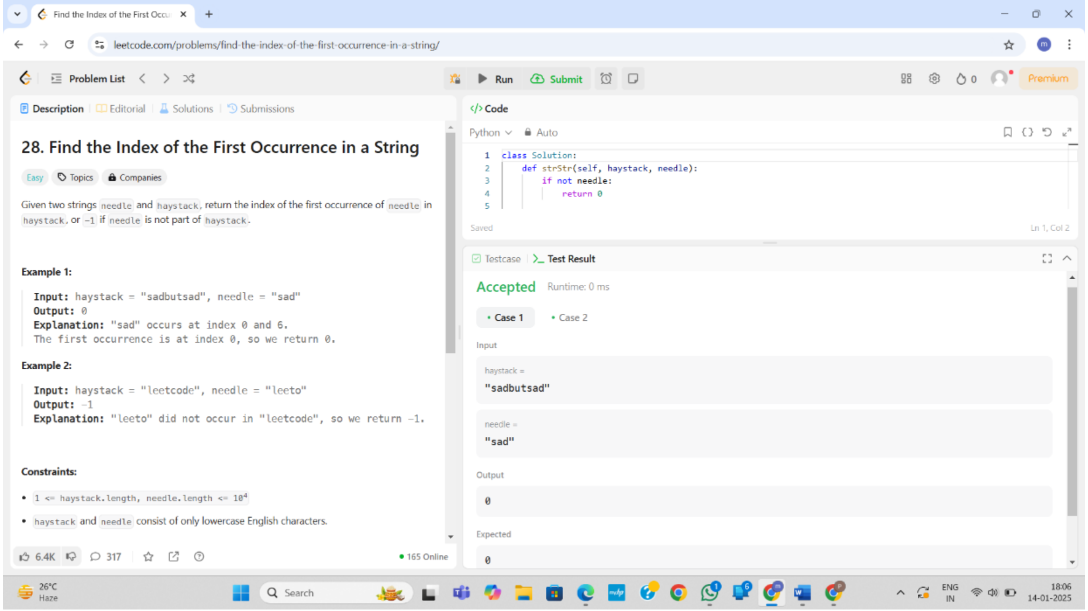
1.Find the Index of the First occurrence in a String

https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string/

CODE:

```
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        if not needle:
            return 0
        haystack_len = len(haystack)
        needle_len = len(needle)
        for i in range(haystack_len - needle_len + 1):
            if haystack[i:i + needle_len] == needle:
                return i
        return -1
```

Results:



28. Find the Index of the First Occurrence in a String

Given two strings `needle` and `haystack`, return the index of the first occurrence of `needle` in `haystack`, or `-1` if `needle` is not part of `haystack`.

Example 1:

```
Input: haystack = "sadbutsad", needle = "sad"
Output: 0
Explanation: "sad" occurs at index 0 and 6.
The first occurrence is at index 0, so we return 0.
```

Example 2:

```
Input: haystack = "leetcode", needle = "leeto"
Output: -1
Explanation: "leeto" did not occur in "leetcode", so we return -1.
```

Constraints:

- `1 <= haystack.length, needle.length <= 10`^{4}
- `haystack` and `needle` consist of only lowercase English characters.

```
class Solution:
    def strStr(self, haystack, needle):
        if not needle:
            return 0

```

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

haystack =
"sadbutsad"

needle =
"sad"

Output

0

Expected

0

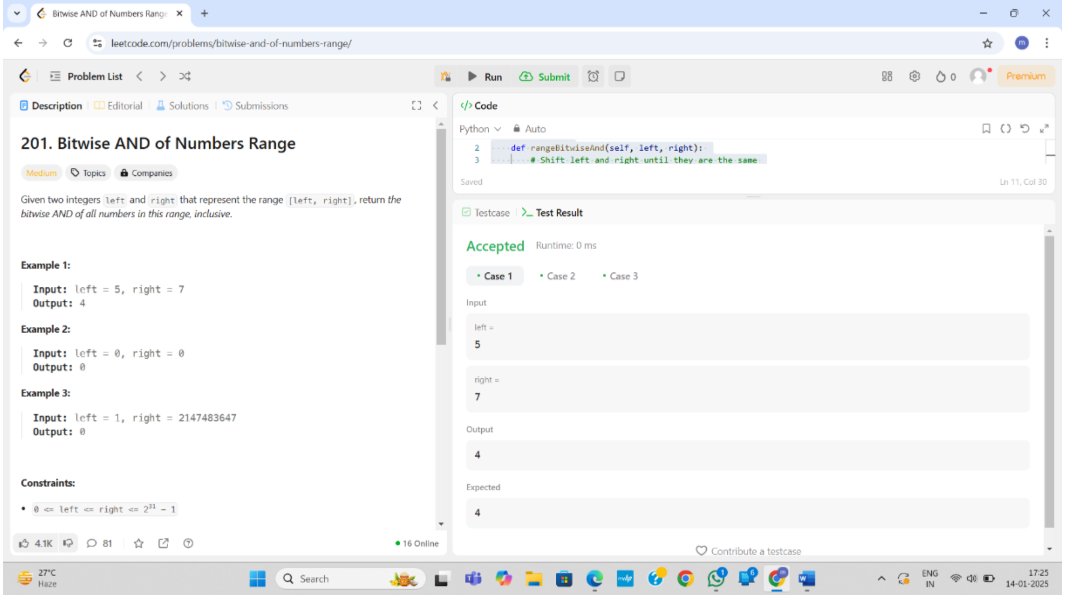
2. Bitwise and of Number Range e

https://leetcode.com/problems/bitwise-and-of-numbers-range/

CODE:

```
class Solution:
    def rangeBitwiseAnd(self, left, right):
        shift = 0
        while left < right:
            left >>= 1
            right >>= 1
            shift += 1
        return left << shift
```

Results:



201. Bitwise AND of Numbers Range

Given two integers `left` and `right` that represent the range `[left, right]`, return *the bitwise AND of all numbers in this range, inclusive.*

Example 1:

```
Input: left = 5, right = 7
Output: 4
```

Example 2:

```
Input: left = 0, right = 0
Output: 0
```

Example 3:

```
Input: left = 1, right = 2147483647
Output: 0
```

Constraints:

- `0 <= left <= right <=` $2^{31} - 1$

```
    def rangeBitwiseAnd(self, left, right):
        # Shift left and right until they are the same
```

Accepted Runtime: 0 ms

Input

left =
5

right =
7

Output

4

Expected

4

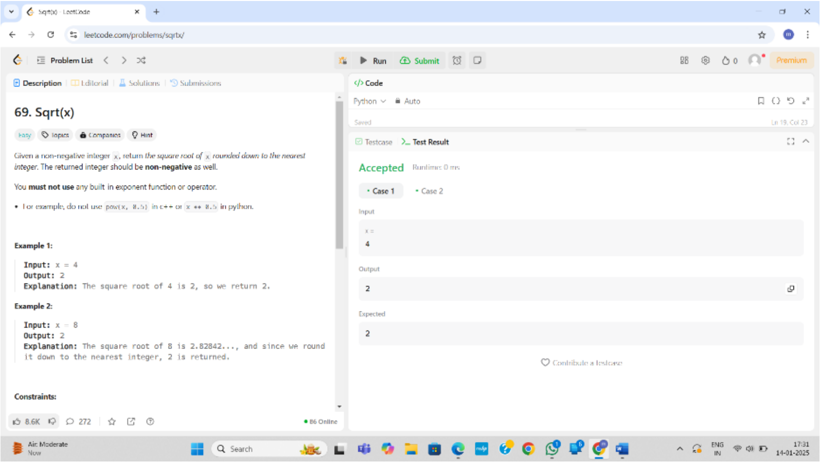
3)Square Root

https://leetcode.com/problems/sqrtx/

CODE:

```
class Solution:
    def mySqrt(self, x):
        if x < 2:
            return x  # For 0 and 1, return x directly
        low, high = 0, x // 2 + 1
        result = 0
        while low <= high:
            mid = (low + high) // 2
            if mid * mid == x:
                return mid  # Perfect square root found
            elif mid * mid < x:
                result = mid  # Update result and move to the right
                low = mid + 1
            else:
                high = mid - 1  # Move to the left
        return result
```

Result:



69. Sqrt(x)

Given a non-negative integer `x`, return *the square root of* `x` *rounded down to the nearest integer*. The returned integer should be **non-negative** as well.

You **must not use** any built in exponent function or operator.

- For example, do not use `pow(x, 0.5)` in c++ or `x ** 0.5` in python.

Example 1:

```
Input: x = 4
Output: 2
Explanation: The square root of 4 is 2, so we return 2.
```

Example 2:

```
Input: x = 8
Output: 2
Explanation: The square root of 8 is 2.82842..., and since we round
it down to the nearest integer, 2 is returned.
```

Constraints:

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

x =
4

Output

2

Expected

2

Contribute a testcase

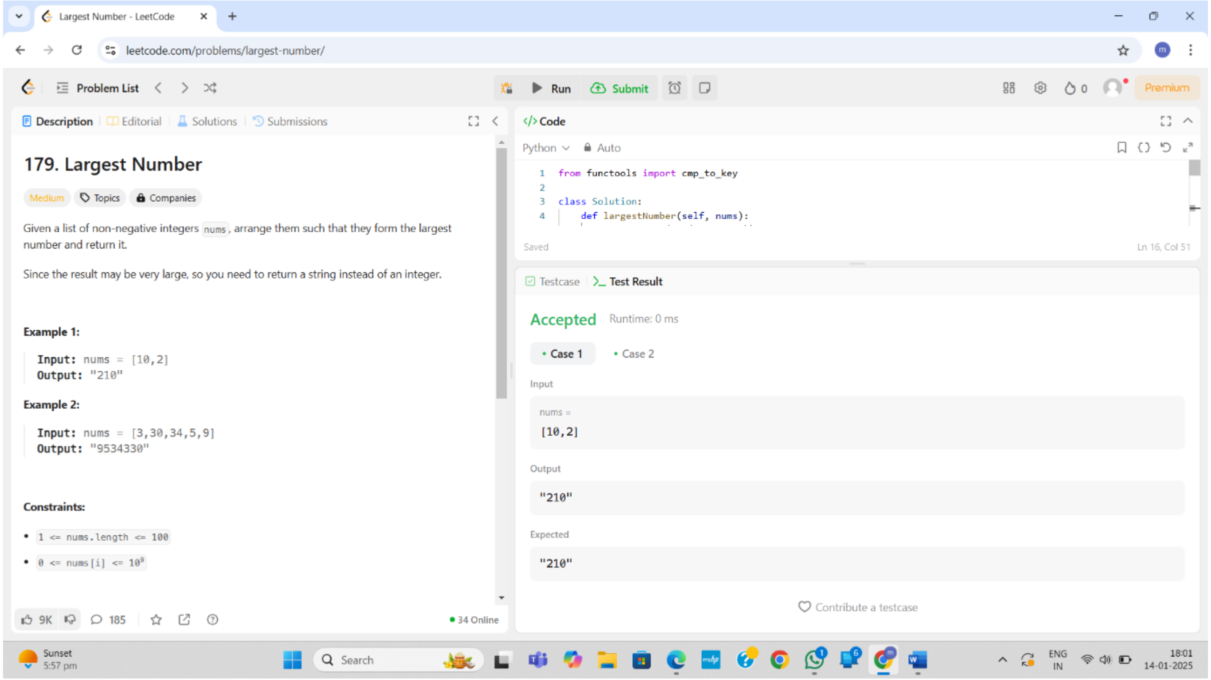
4. largest-number

https://leetcode.com/problems/largest-number/

CODE:

```
from functools import cmp_to_key
class Solution:
    def largestNumber(self, nums):
        nums = list(map(str, nums))
        def compare(x, y):
            if x + y > y + x:
                return -1
            elif x + y < y + x:
                return 1
            else:
                return 0
        nums.sort(key=cmp_to_key(compare))
        result = ''.join(nums)
        return '0' if result[0] == '0' else result
```

Result:



179. Largest Number

Given a list of non-negative integers `nums`, arrange them such that they form the largest number and return it.

Since the result may be very large, so you need to return a string instead of an integer.

Example 1:

```
Input: nums = [10,2]
Output: "210"
```

Example 2:

```
Input: nums = [3,30,34,5,9]
Output: "9534330"
```

Constraints:

- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 10`^{9}

```
from functools import cmp_to_key

class Solution:
    def largestNumber(self, nums):
```

Accepted Runtime: 0 ms

Input

```
nums =
[10,2]
```

Output

```
"210"
```

Expected

```
"210"
```

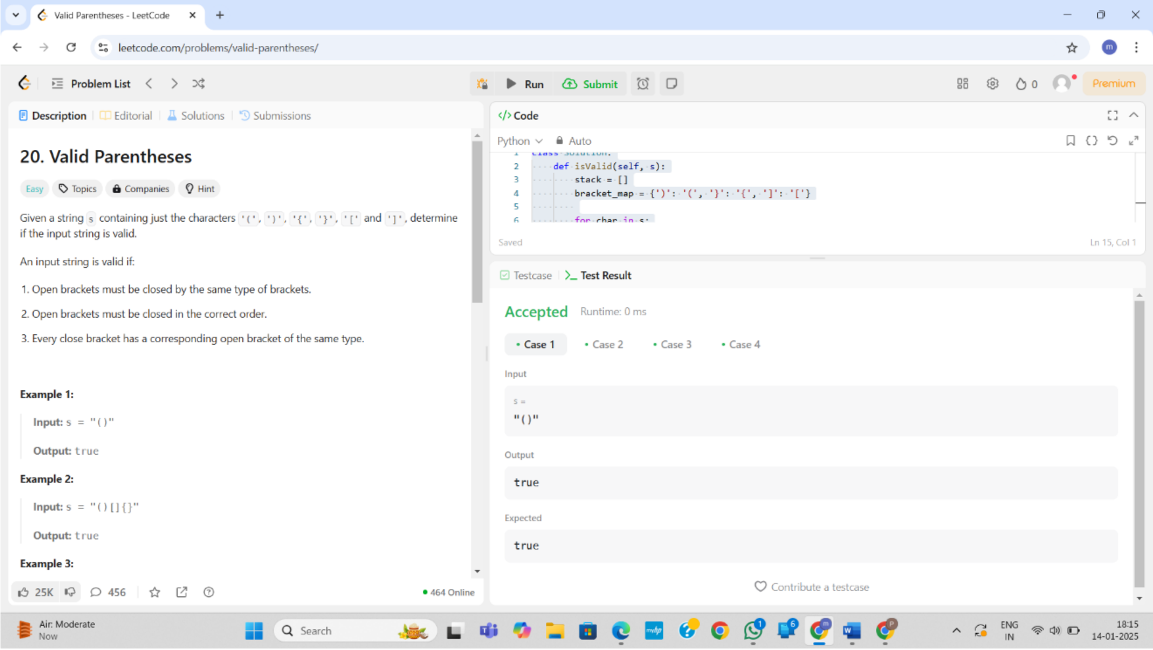
5. Valid Parentheses

https://leetcode.com/problems/valid-parentheses/description/

CODE:

```
class Solution:
    def isValid(self, s):
        stack = []
        bracket_map = {')': '(', '}': '{', ']': '['}
        for char in s:
            if char in bracket_map:
                top_element = stack.pop() if stack else '#'
                if bracket_map[char] != top_element:
                    return False
            else:
                stack.append(char)
        return not stack
```

Result:



20. Valid Parentheses

Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

Example 1:

Input: s = "()"

Output: true

Example 2:

Input: s = "()[]{}"

Output: true

Example 3:

```
class Solution:
    def isValid(self, s):
        stack = []
        bracket_map = {')': '(', '}': '{', ']': '['}

        for char in s:
```

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3 • Case 4

Input

s =

"()"

Output

true

Expected

true

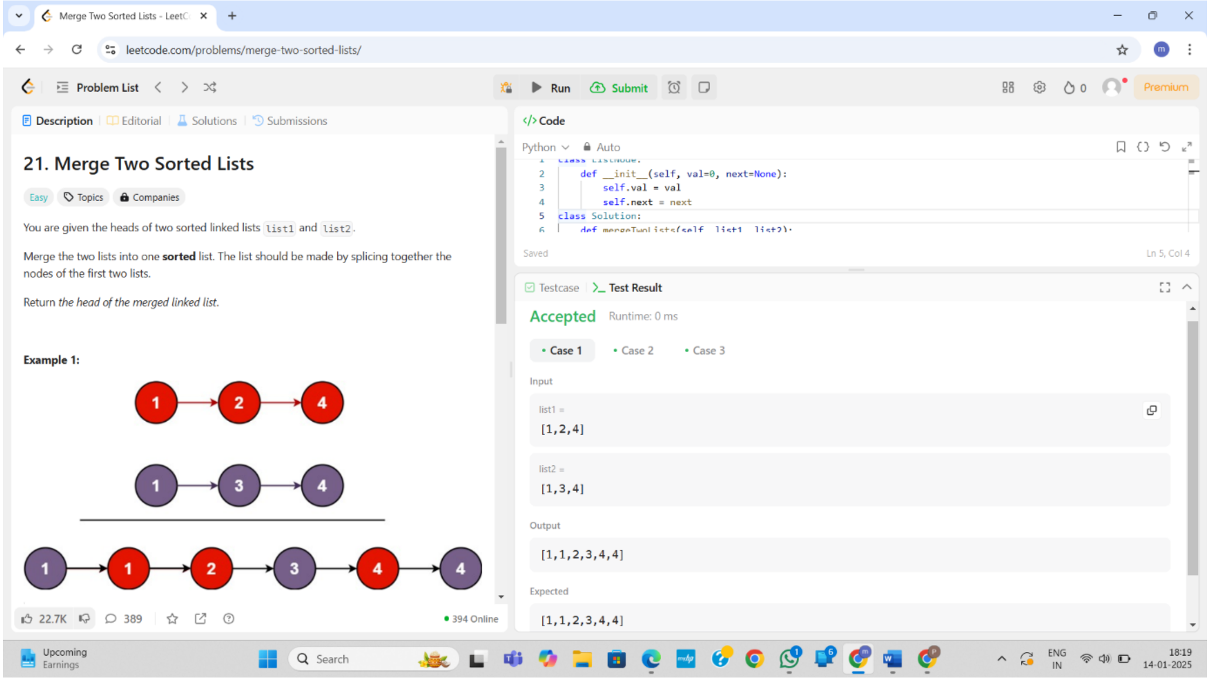
6. merge-two-sorted-lists

https://leetcode.com/problems/merge-two-sorted-lists/

CODE:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
    def mergeTwoLists(self, list1, list2):
        dummy = ListNode()
        current = dummy
        while list1 and list2:
            if list1.val < list2.val:
                current.next = list1
                list1 = list1.next
            else:
                current.next = list2
                list2 = list2.next
            current = current.next
        if list1:
            current.next = list1
        elif list2:
            current.next = list2
        return dummy.next
```

Result:



21. Merge Two Sorted Lists

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 ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
    def mergeTwoLists(self, list1, list2):
```

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

list1 =

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

list2 =

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

Output

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

Expected

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

7. remove-duplicates-from-sorted-list

https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/

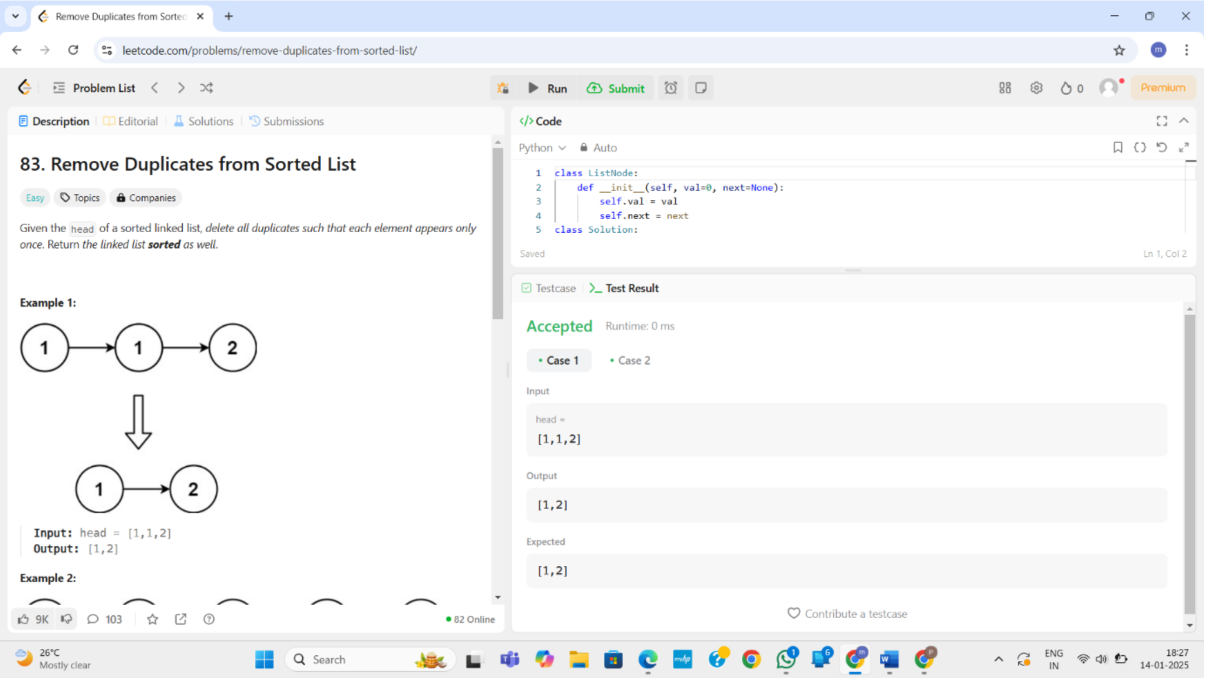
CODE:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
    def deleteDuplicates(self, head):
        current = head

        while current and current.next:
            if current.val == current.next.val:
                # Skip the next node if it's a duplicate
                current.next = current.next.next
            else:
                current = current.next

        return head
```

Result:



leetcode.com/problems/remove-duplicates-from-sorted-list/

83. Remove Duplicates from Sorted List

Given the `head` of a sorted linked list, *delete all duplicates such that each element appears only once*. Return *the linked list **sorted** as well.*

Example 1:

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

Example 2:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
```

Saved

Ln 1, Col 2

Accepted Runtime: 0 ms

Input

head =
[1,1,2]

Output

[1,2]

Expected

[1,2]

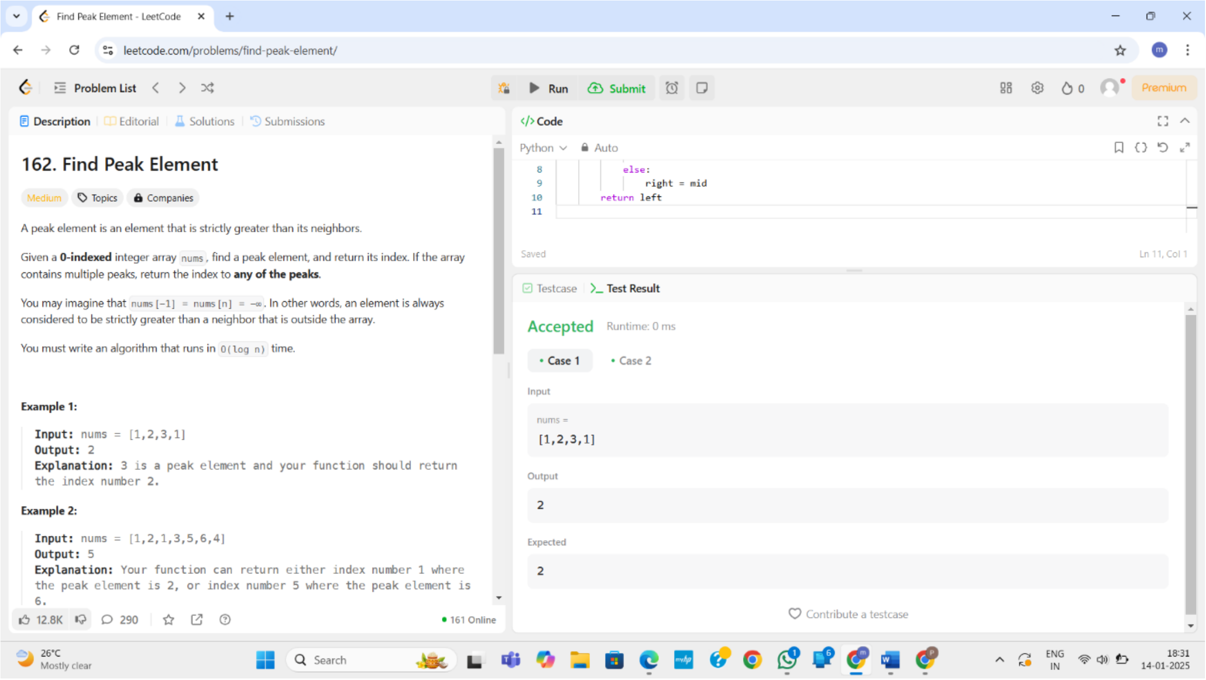
8. find-peak-element

https://leetcode.com/problems/find-peak-element/description/

CODE:

```
class Solution:
    def findPeakElement(self, nums):
        left, right = 0, len(nums) - 1
        while left < right:
            mid = (left + right) // 2
            if nums[mid] < nums[mid + 1]:
                left = mid + 1
            else:
                right = mid
        return left
```

Result:



162. Find Peak Element

A peak element is an element that is strictly greater than its neighbors.

Given a **0-indexed** integer array `nums`, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that `nums[-1] = nums[n] = -∞`. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

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

Example 1:

```
Input: nums = [1,2,3,1]
Output: 2
Explanation: 3 is a peak element and your function should return the index number 2.
```

Example 2:

```
Input: nums = [1,2,1,3,5,6,4]
Output: 5
Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.
```

```
            else:
                right = mid
        return left
```

Saved

Ln 11, Col 1

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =

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

Output

```
2
```

Expected

```
2
```

9. Binary-tree-inorder-traversal

https://leetcode.com/problems/binary-tree-inorder-traversal/description/

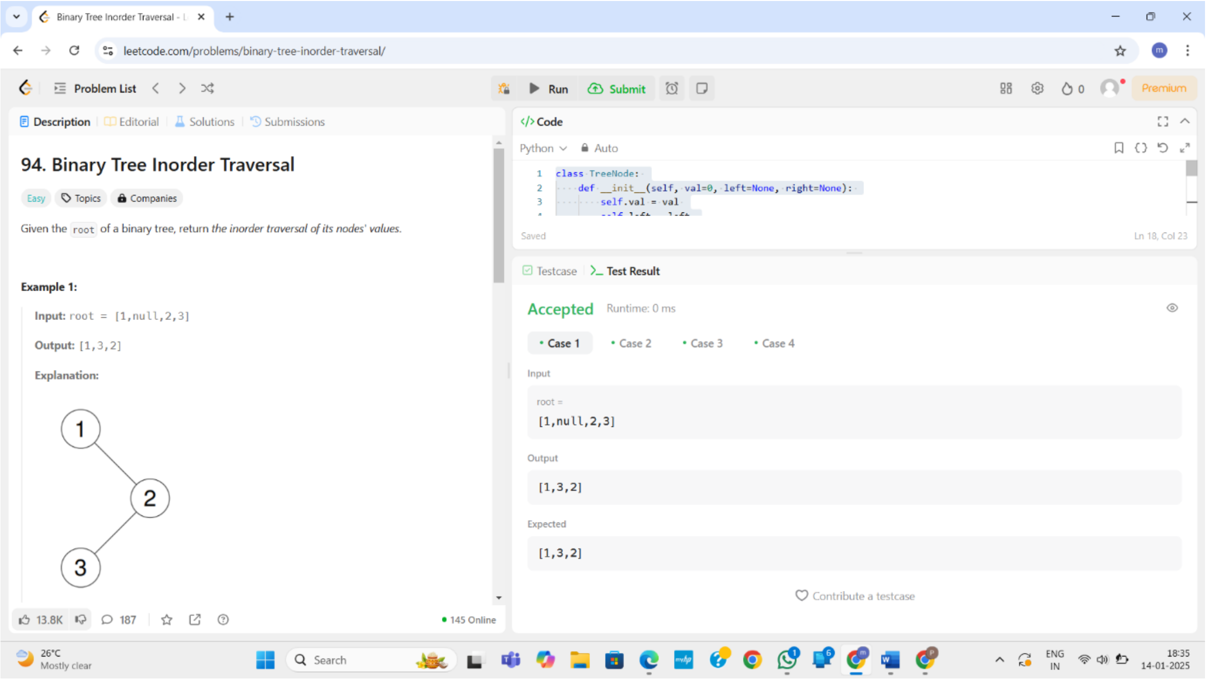
CODE:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def inorderTraversal(self, root):
        result = []
        def inorder(node):
            if node:
                inorder(node.left)
                result.append(node.val)
                inorder(node.right)

        inorder(root)
        return result
```

Result:



94. Binary Tree Inorder Traversal

Easy · Topics · Companies

Given the `root` of a binary tree, return *the inorder traversal of its nodes' values*.

Example 1:

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

Output: `[1,3,2]`

Explanation:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
```

Saved

Ln 18, Col 23

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3 • Case 4

Input

root =

```
[1,null,2,3]
```

Output

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

Expected

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

Contribute a testcase

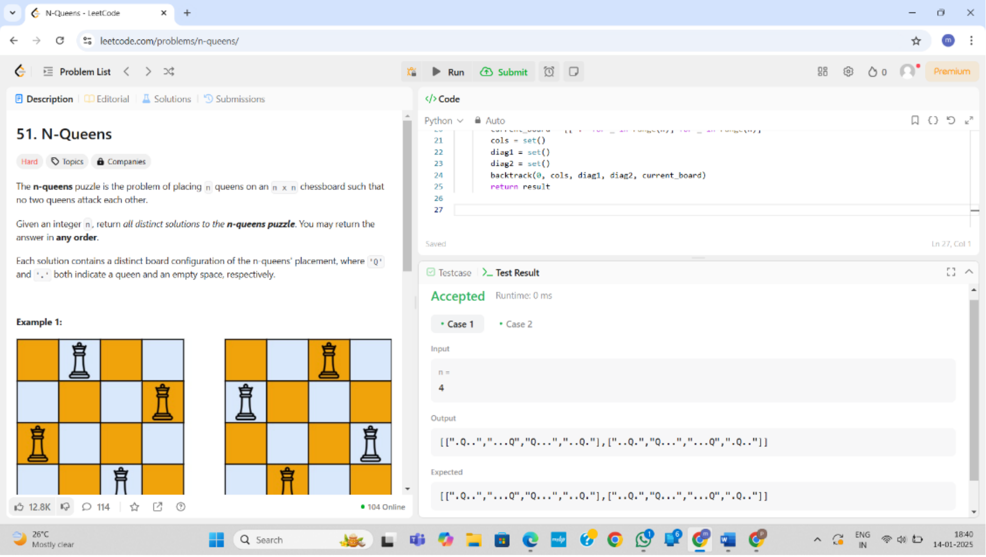
10. N-queens

https://leetcode.com/problems/n-queens/description/

CODE:

```
class Solution:
    def solveNQueens(self, n):
        result = []
        def backtrack(row, cols, diag1, diag2, current_board):
            if row == n:
                result.append(["".join(row) for row in current_board])
                return
            for col in range(n):
                if col in cols or (row - col) in diag1  or (row + col) in diag2:
                    continue
                cols.add(col)
                diag1.add(row - col)
                diag2.add(row + col)
                current_board[row][col] = 'Q'
                backtrack(row + 1, cols, diag1, diag2, current_board)
                cols.remove(col)
                diag1.remove(row - col)
                diag2.remove(row + col)
                current_board[row][col] = '.'
        current_board = [["." for _ in range(n)] for _ in range(n)]
        cols = set()
        diag1 = set()
        diag2 = set()
        backtrack(0, cols, diag1, diag2, current_board)
        return result
```

Result:



51. N-Queens

The **n-queens** puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other.

Given an integer n, return *all distinct solutions to the* ***n-queens puzzle***. You may return the answer in **any order**.

Each solution contains a distinct board configuration of the n-queens' placement, where 'Q' and '.' both indicate a queen and an empty space, respectively.

Example 1:

```
        cols = set()
        diag1 = set()
        diag2 = set()
        backtrack(0, cols, diag1, diag2, current_board)
        return result
```

Accepted Runtime: 0 ms

Input

n =
4

Output

```
[[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]
```

Expected

```
[[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]
```