DESIGN AND ANALYSIS OF ALGORITHMS

Name:P.Vamshi
Roll No: 2211CS020411
Section: AIML-Epsilon

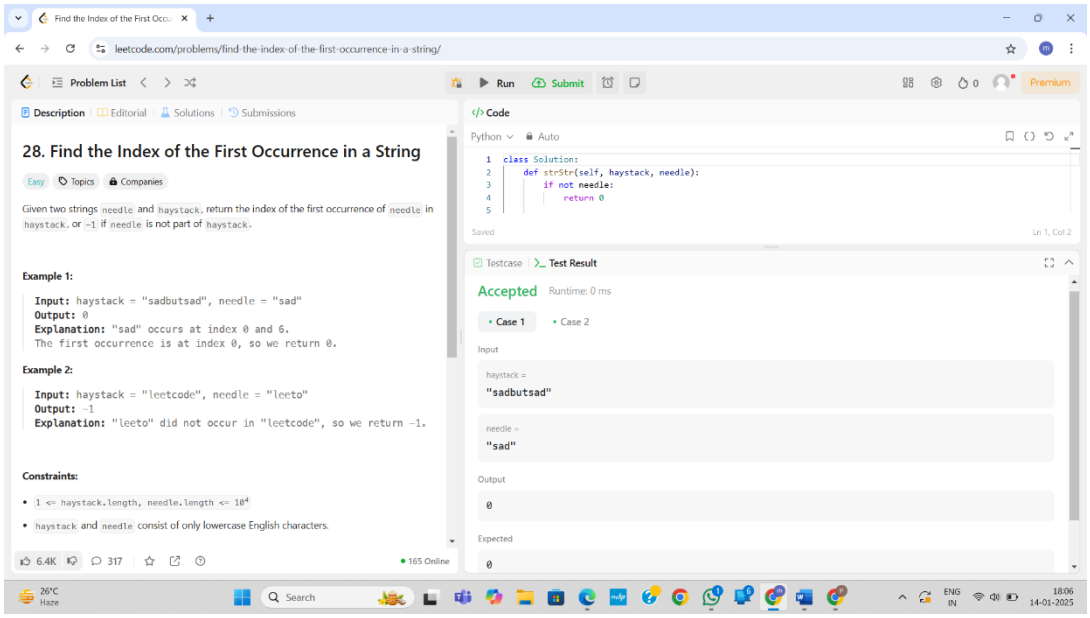
1.Find the Index of the First occurrence in a String

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:



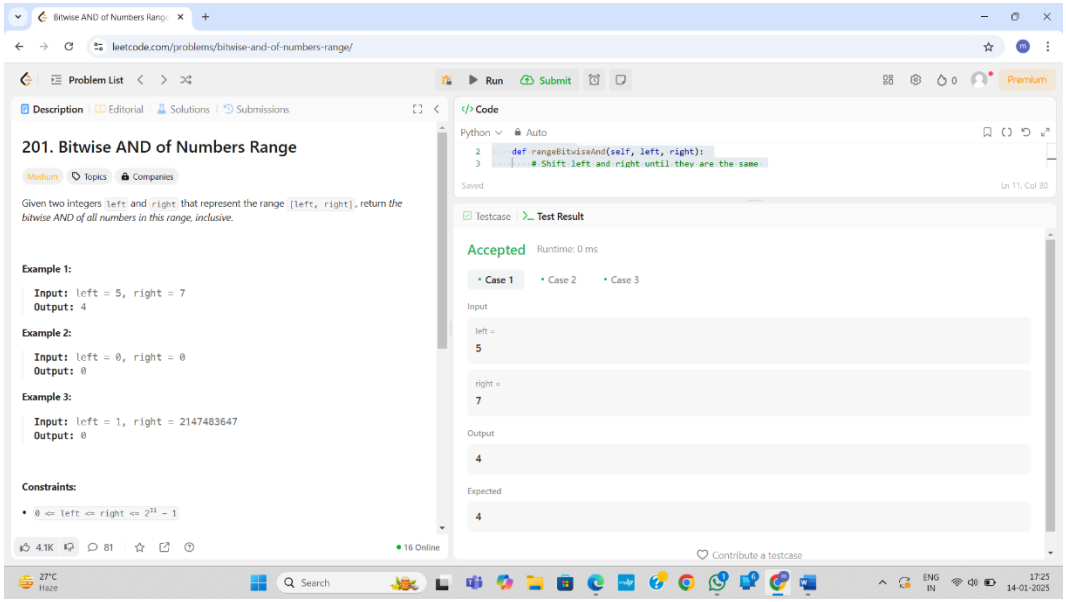
2. Bitwise and of Number Range e

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:



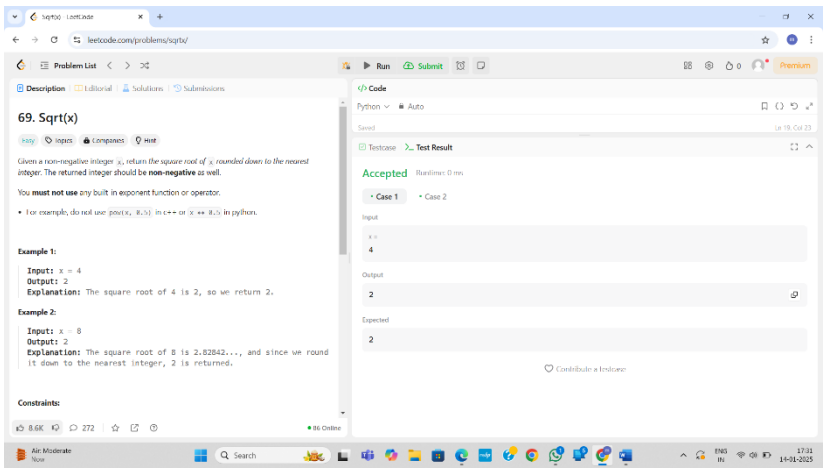
3)Square Root

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:



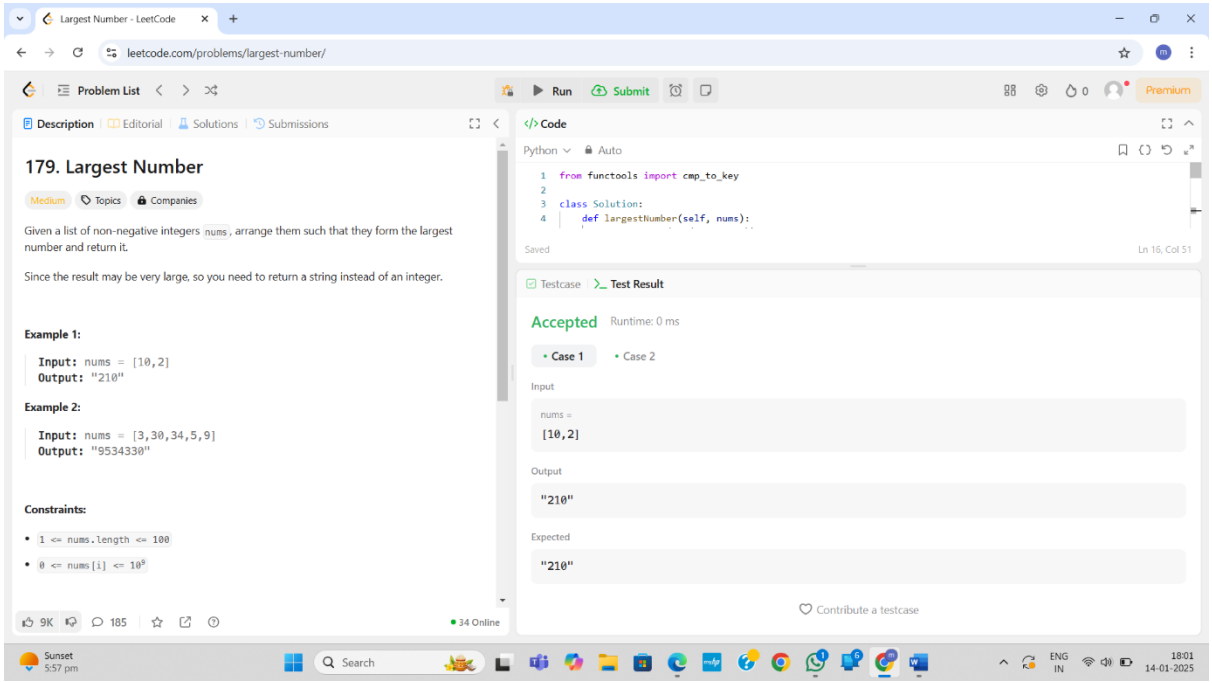
4. largest-number

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:



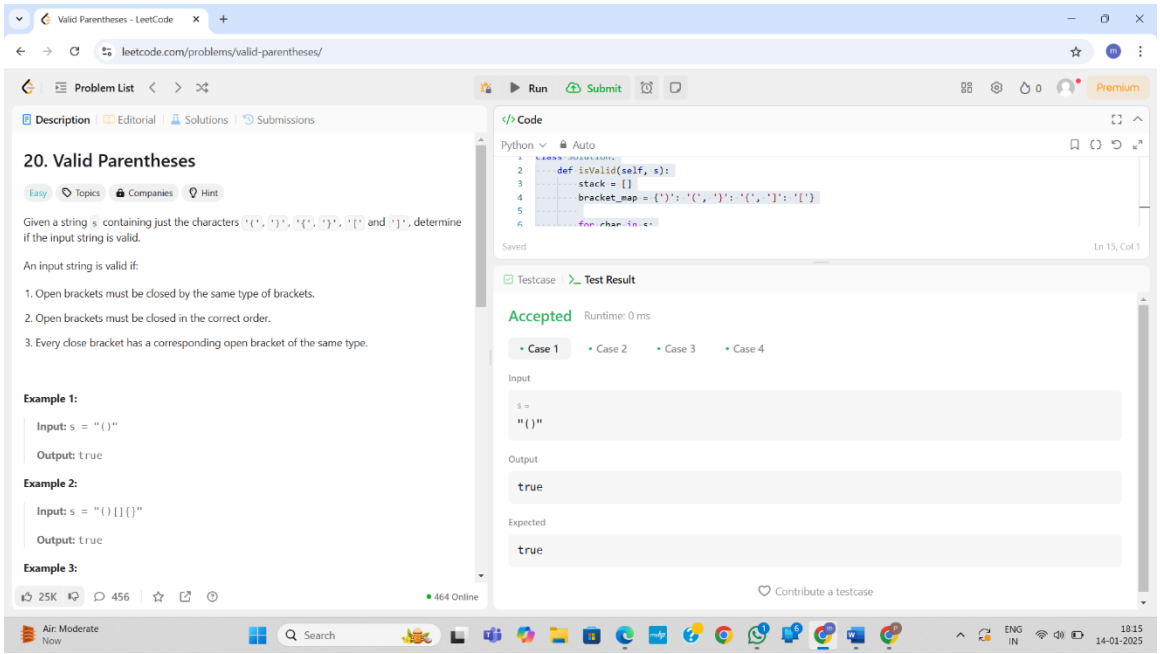
5. Valid Parentheses

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:



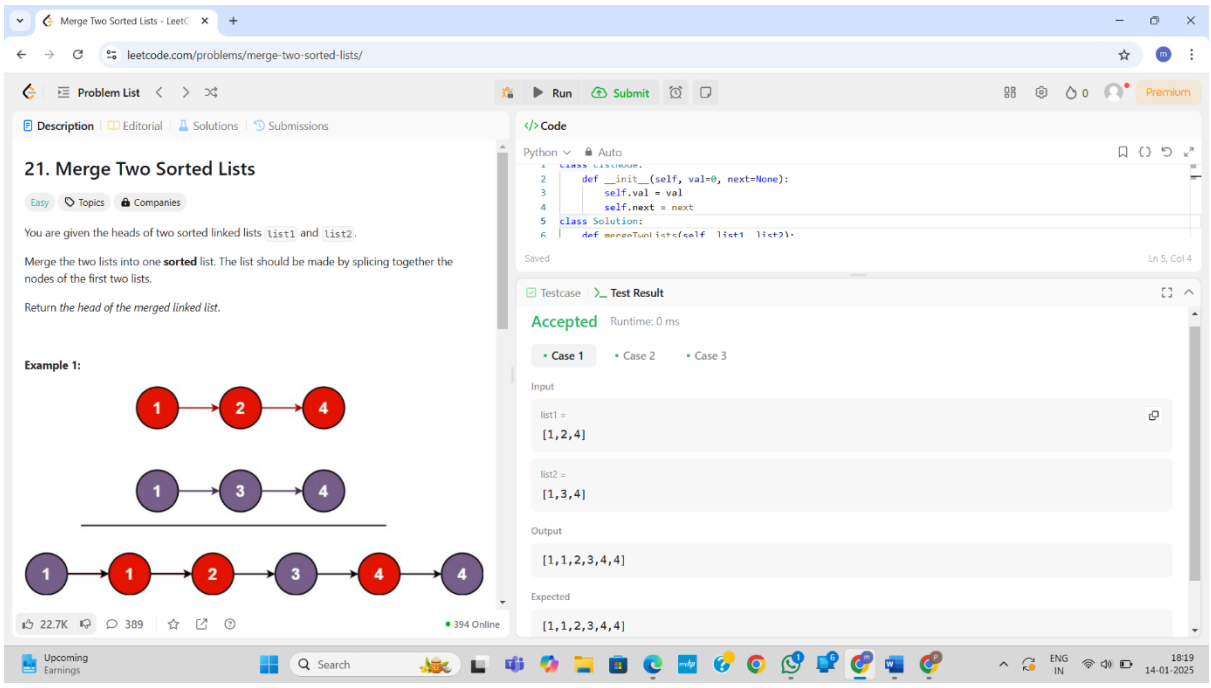
6. merge-two-sorted-lists

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:



7. remove-duplicates-from-sorted-list

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

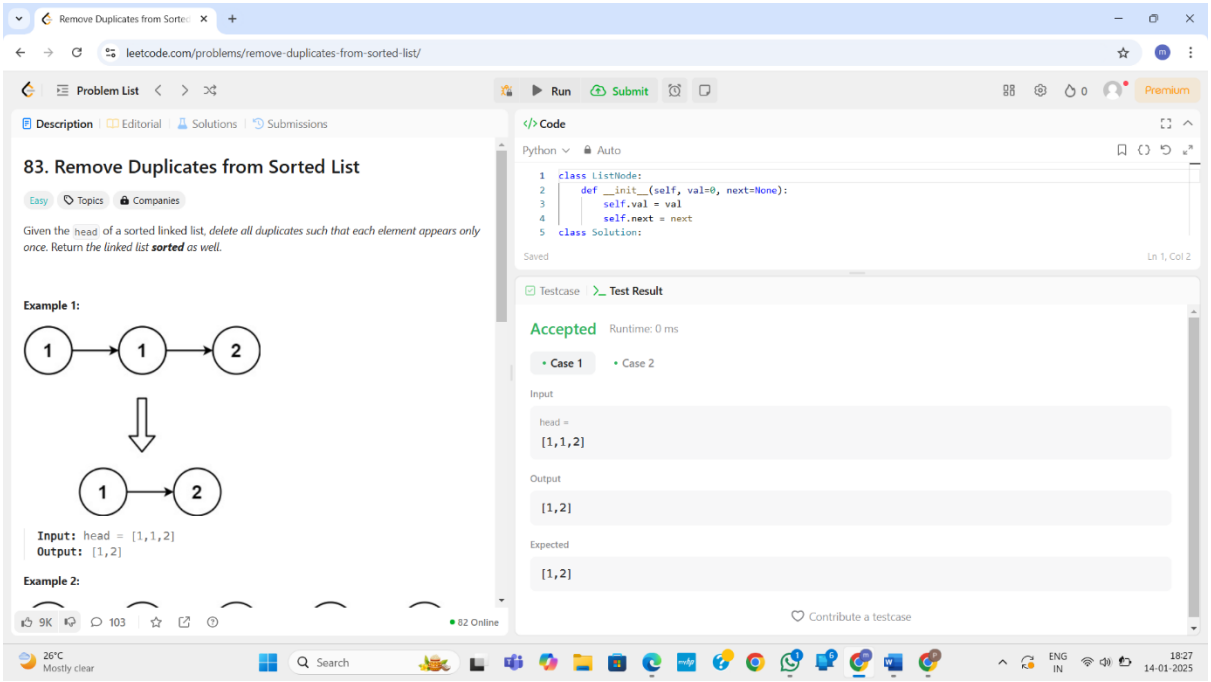
CODE:

```
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:



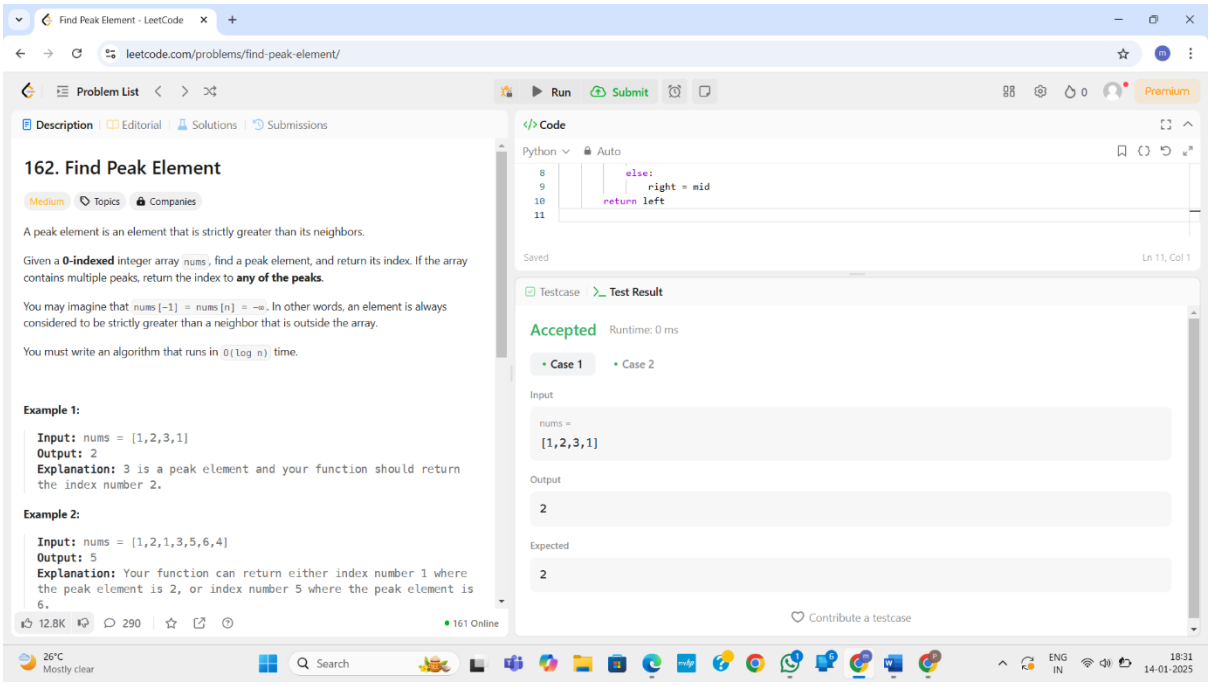
8. find-peak-element

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:



9. Binary-tree-inorder-traversal

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

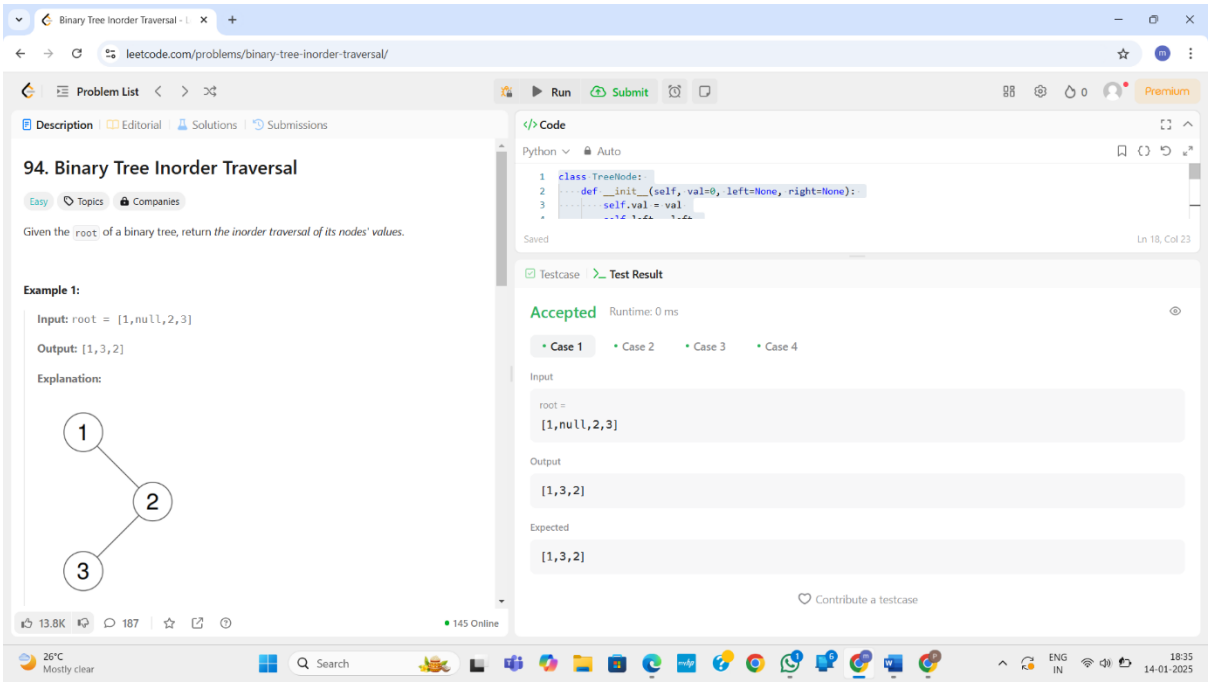
CODE:

```
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:



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: