RAMAIAH INSTITUTE OF TECHNOLOGY

MSRIT NAGAR, BENGALURU,560054

A Report on

[Design and Analysis of Algorithms]

Submitted in partial fulfilment of the OTHER COMPONENT requirements as a part of the DAA LAB subject with code IS46 for the IV Semester of degree of ***Bachelor of Engineering in Information Science and Engineering***

Submitted by

Candidate Names

PRATHIBHA A

1MS23IS409

Under the Guidance of

Faculty In charge

Shivanandha

Dept. of ISE

Department of Information Science and Engineering

Ramaiah Institute of Technology

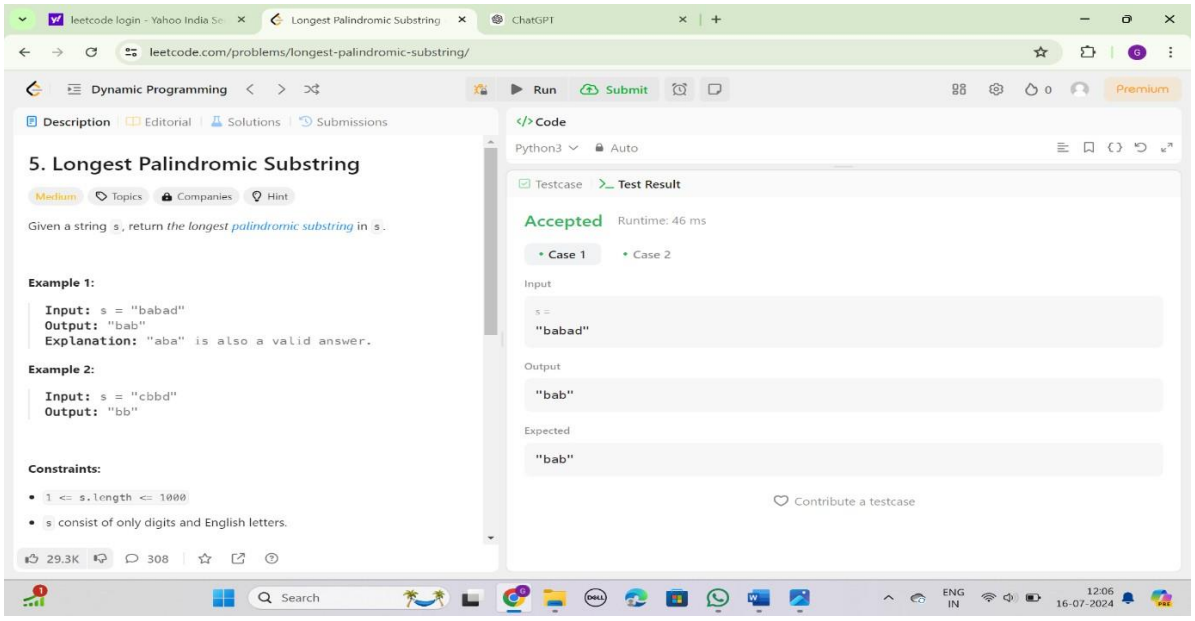
2023– 2024

1.Dynamic programming

1.1 longest palindrome substringCode:

```
class Solution:
    def longestPalindrome(self, s) :
        if len(s) <= 1:
            return s
        Max_Len=1
        Max_Str=s[0]
        for i in range(len(s)-1):
            for j in range(i+1,len(s)):
                if j-i+1 > Max_Len and s[i:j+1] == s[i:j+1][::-1]:
                    Max_Len = j-i+1
                    Max_Str = s[i:j+1]
    return Max_Str
```

OUTPUT:

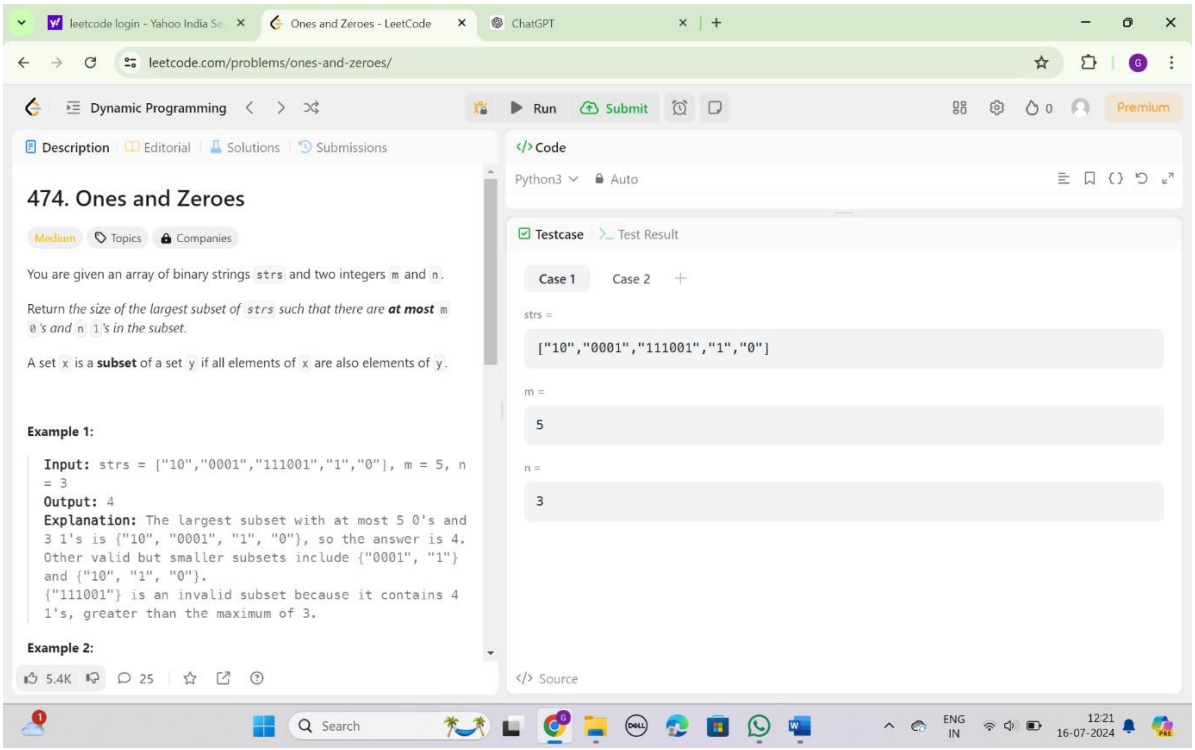


1.2 ones and zeros

Code:

```
class Solution:
  def findMaxForm(self, strs, m, n):
      xy = [[s.count("0"), s.count("1")] for s in strs]
      def dp(mm, nn, kk):
          if mm < 0 or nn < 0: return -float("inf")
          if kk == len(strs): return 0
          x, y = xy[kk]
          return max(1 + dp(mm-x, nn-y, kk + 1), dp(mm, nn, kk + 1))
      return dp(m, n, 0)
```

OUTPUT:

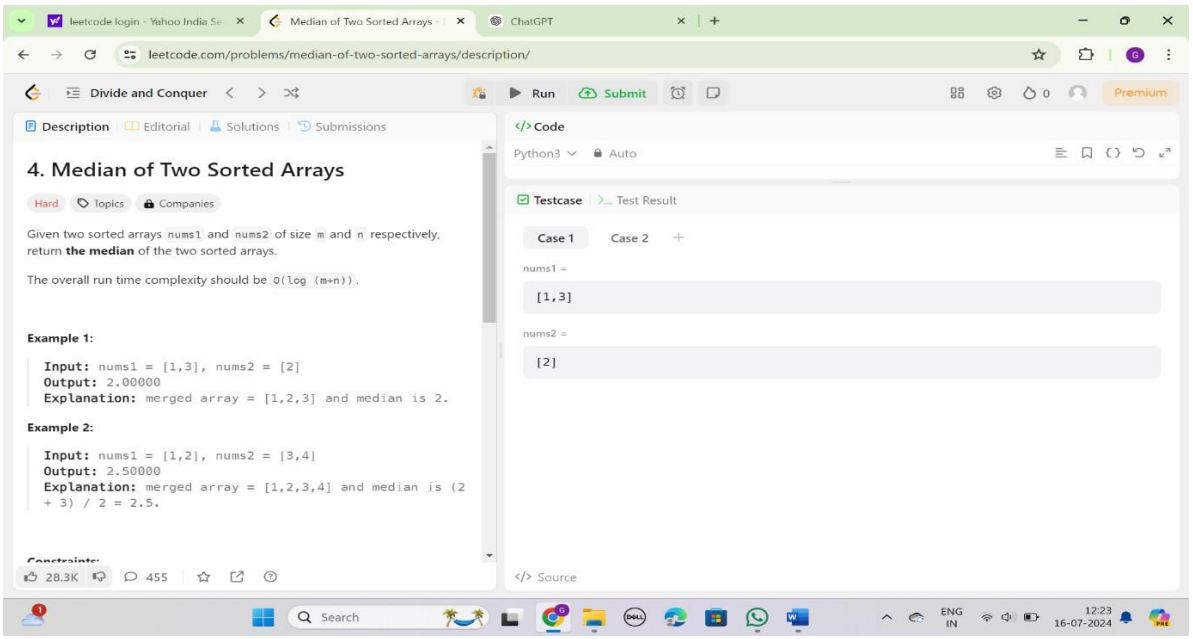


2.devide and conquare

2.1 median of two sorted arrayCode:

```
class Solution:
 def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:
    len1 = len(nums1)
    len2 = len(nums2)
    total_len = len1+len2
    lst = nums1+nums2
    lst.sort()
    if total_len % 2 == 0:
        median = (lst[total_len // 2 - 1] + lst[total_len // 2]) / 2
    else:
        median = lst[total_len // 2]
    return median
```

OUTPUT:

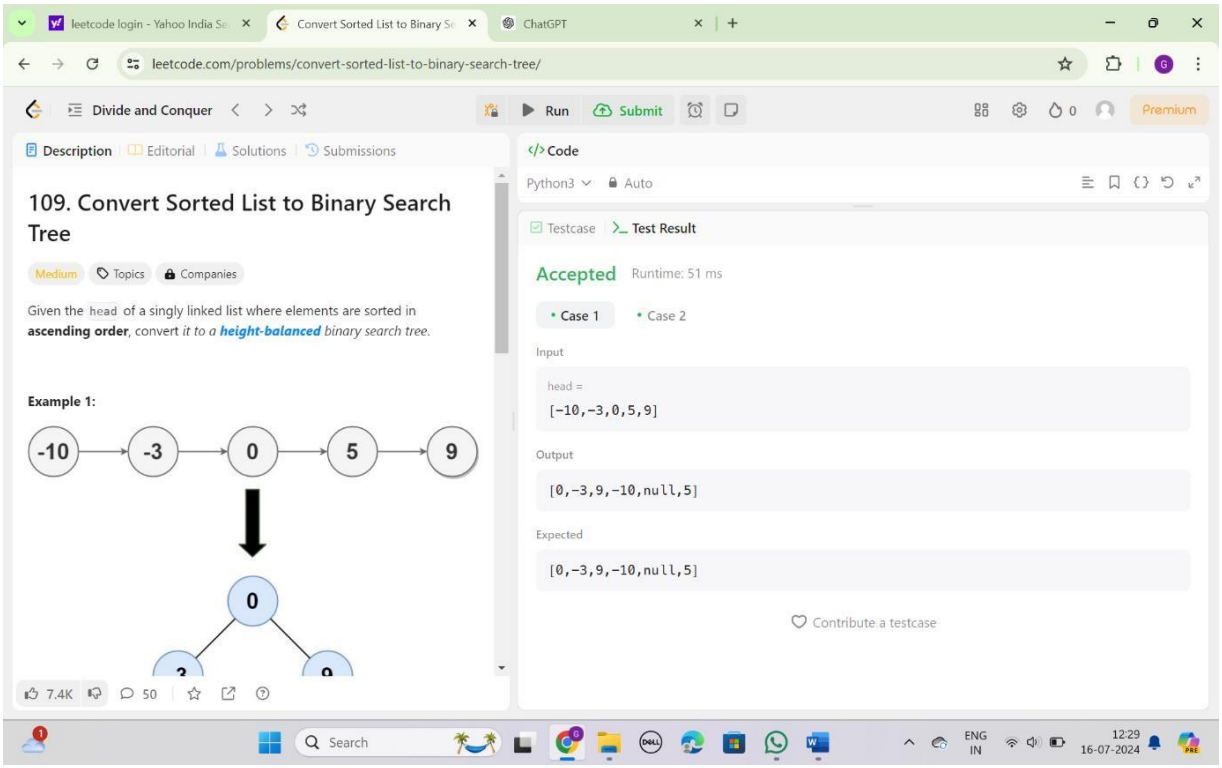


2.2 convert sorted list to binary search

Code:

```
class Solution:
    def sortedListToBST(self, head: Optional[ListNode]) -> Optional[TreeNode]:
        def middle(curr = None):
            if curr:
                prev = slow = fast =
                currwhile fast and
                fast.next:
                    prev =  slow
                    slow =
                    slow.next
                    fast = fast.next.next
                return prev
        def helper(curr=None):
            if curr:
                pre_mid = middle(curr)
                if not pre_mid.next:
                    return TreeNode(pre_mid.val)
                parent = pre_mid.next
                pre_mid.next = None
                root = TreeNode(parent.val)
                root.right = helper(parent.next)
                root.left = helper(curr)
                return root
        return helper(head)
```

OUTPUT:



3. Greedy Technique

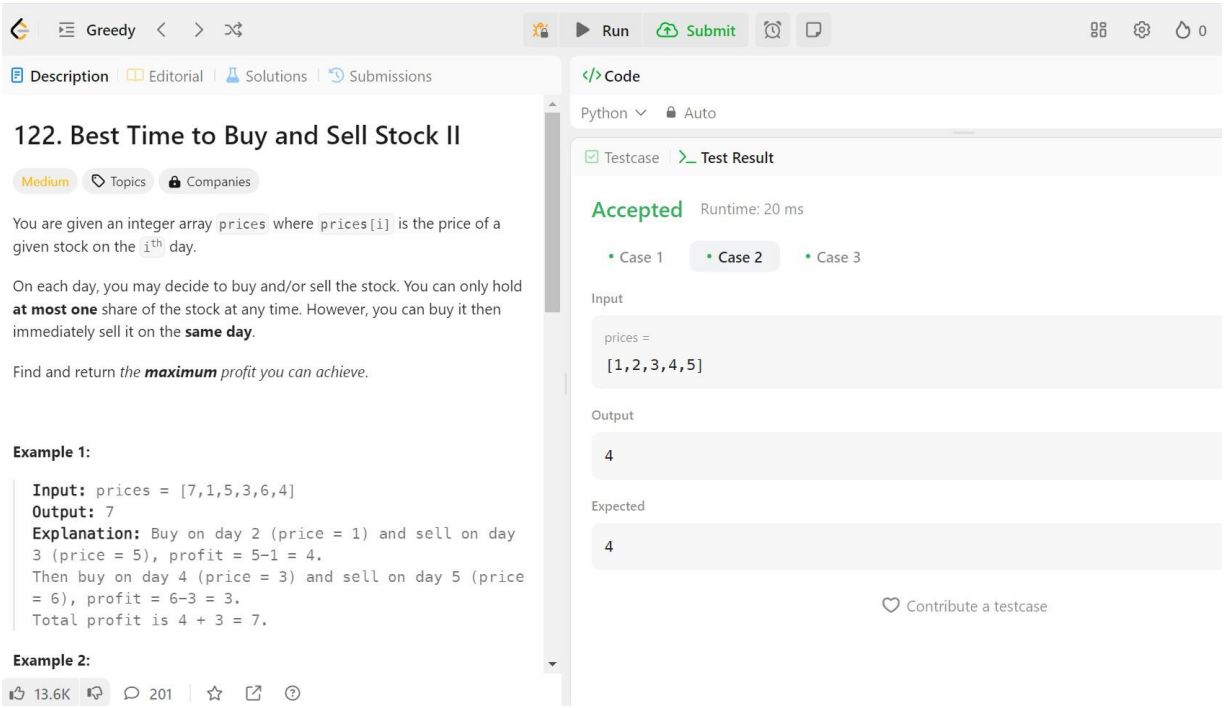
3.1 Best time to buy and sell stock

Code:

```
class Solution(object):
    def maxProfit(self, prices):
        max = 0
        start = prices[0]
        len1 = len(prices)
        for i in range(0 ,
            len1):if start <
            prices[i]:
                max += prices[i] - start
```

```
    start = prices[i]
  return max
```

OUTPUT:



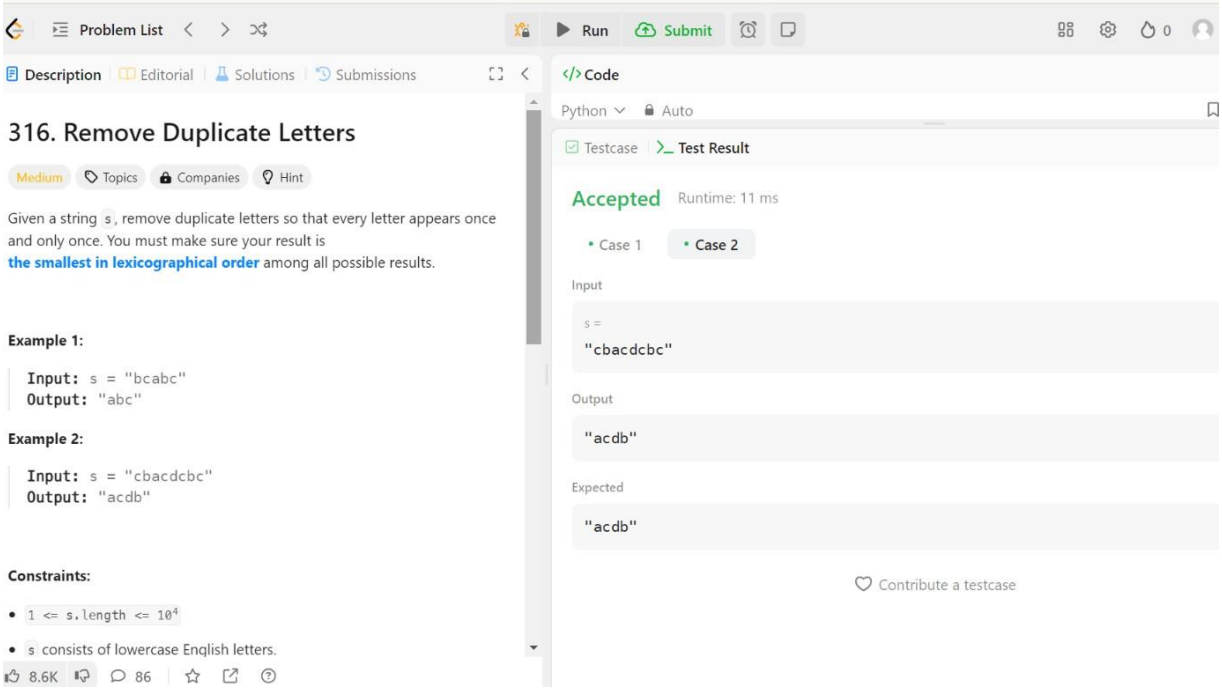
3.2 remove duplicate letters

Code:

```
class Solution:
  def removeDuplicateLetters(self, s):
    last_index = {}
    for k, v in enumerate(s):
      last_index[v] = k

    stack = []
    visited = set()

    for i in
      range(len(s)):if
      s[i] in visited:
```

```
            continue

        while stack and stack[-1] > s[i] and last_index[stack[-1]] >
            i:visited.remove(stack[-1])
            stack.pop()

        stack.append(s[i])
        visited.add(s[i])

    return "".join(stack)
```

OUTPUT:



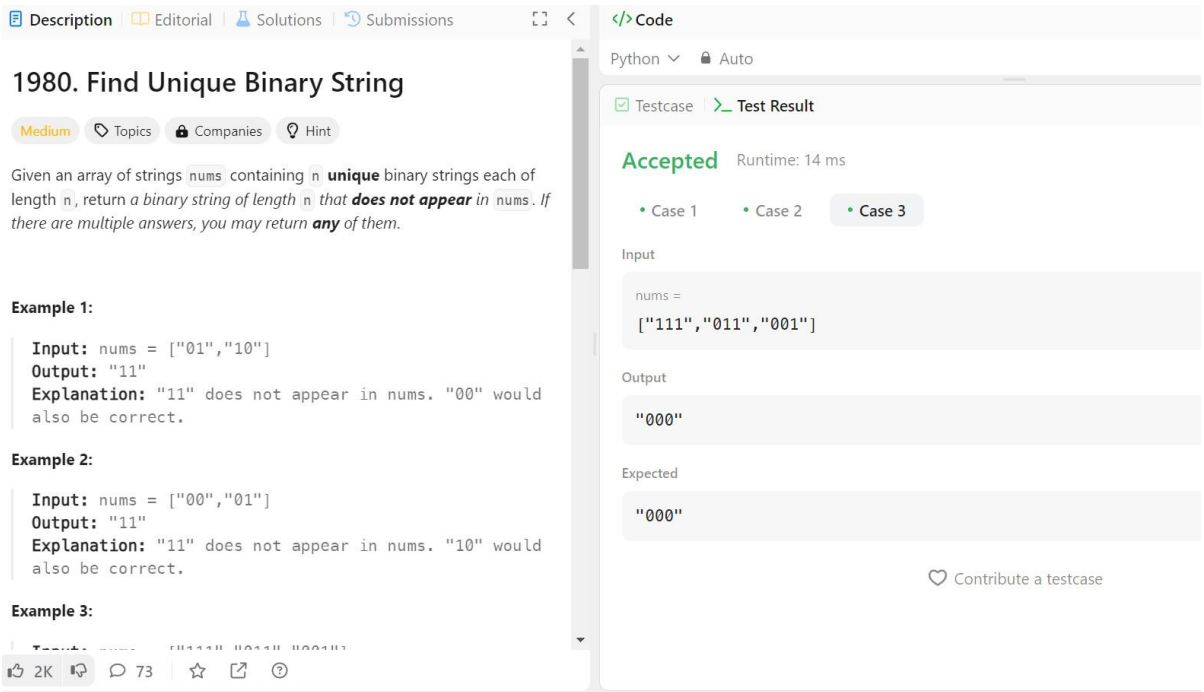
4. Back Tracking:

4.1 Find unique binary string

Code:

```
class Solution(object):
  def findDifferentBinaryString(self, nums):
      result = ""

      for i in range(len(nums)):
        result += '1' if nums[i][i] == '0' else '0'

      return result
```

OUTPUT:



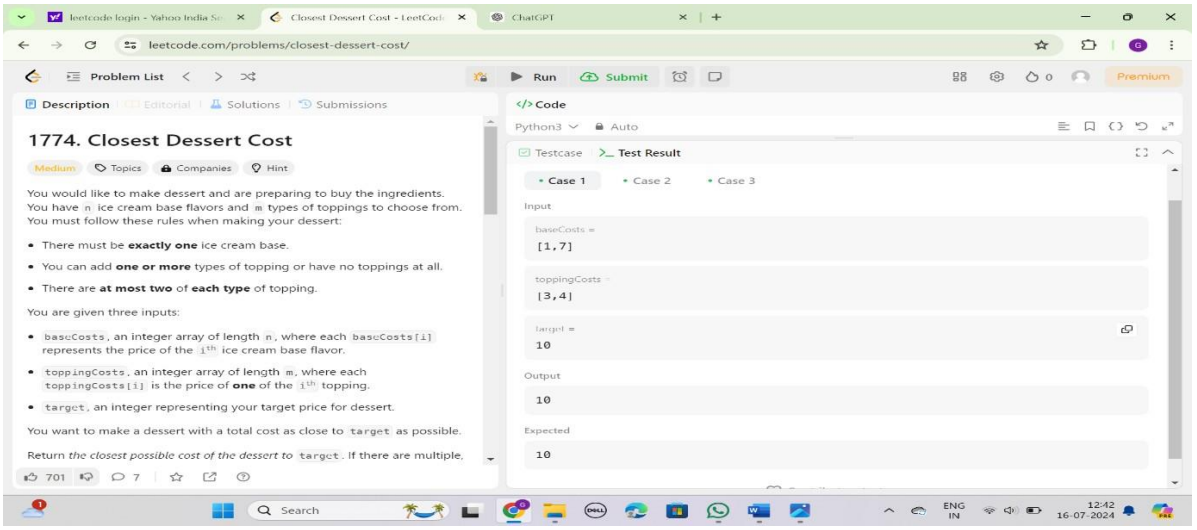
4.2 Closest desert

costCode:

```
class Solution:
  def closestCost(self, baseCosts: List[int], toppingCosts: List[int], target: int) -> int:
      ans = float('inf')
```

```
    # memo to accelerate
    @lru_cache(None)
    def helper(idx, comb):
        nonlocal ans
        if abs(comb - target) == abs(ans - target):
            ans = min(ans,comb)
        if abs(comb - target) < abs(ans - target):ans
            = comb
        for i in range(idx,
            len(toppingCosts)):helper(i + 1,
            comb)
            helper(i + 1, comb + toppingCosts[i])
            helper(i + 1, comb + 2 * toppingCosts[i])

    for bc in baseCosts:
        helper(0, bc)

    return ans
```

OUTPUT:

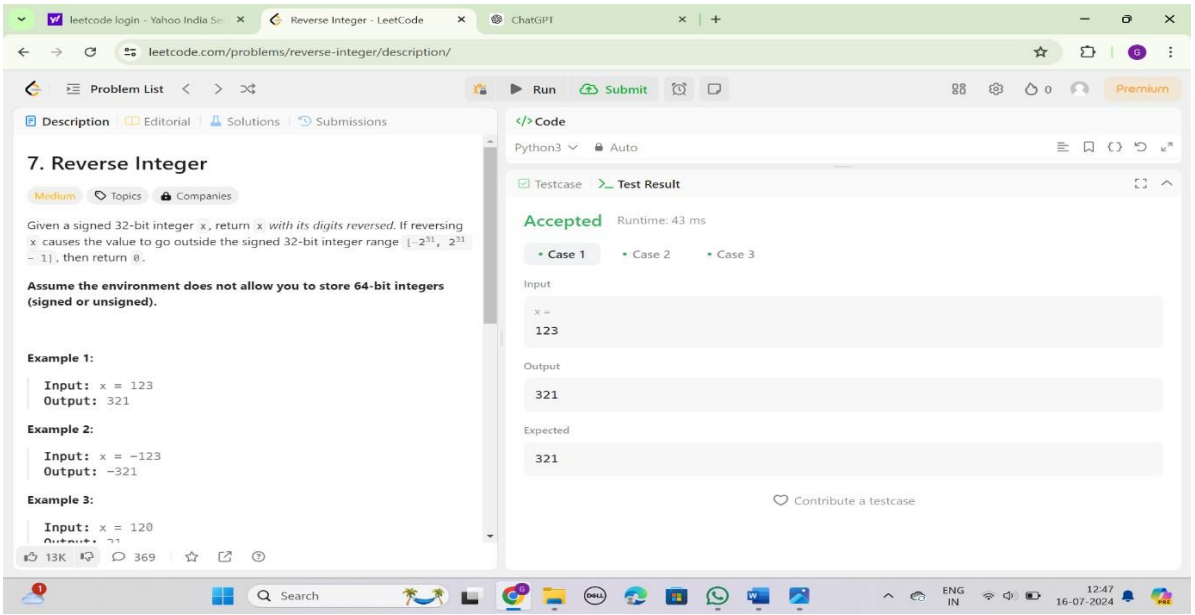


5.Brute Force:

5.1 Reverse integerCode:

```
class Solution:
    def reverse(self, x: int) ->
        int:res = 0
        if x < 0:
            res = int(str(x)[1:][::-1]) * -
        1else:
            res = int(str(x)[::-1])

        if res > 2 ** 31 - 1 or res < -2 ** 31:
            return 0

        return res
```

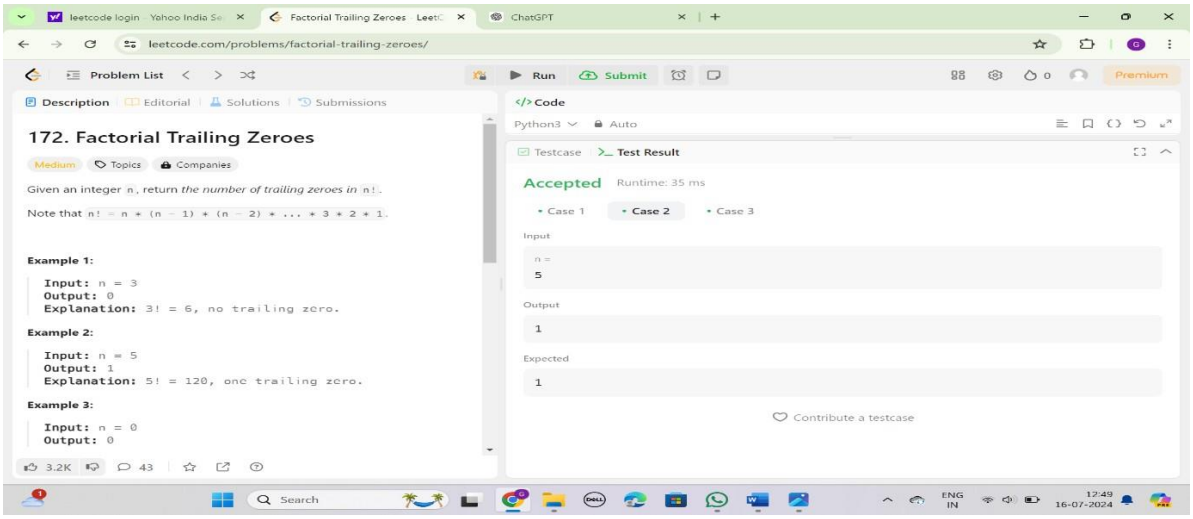
OUTPUT:



5.2 Factorial trailing zerosCode:

```
class Solution:
    def trailingZeroes(self, n: int) ->
        int:fact = 1
        for i in range(1, n+1):
            fact = fact * i
        sys.set_int_max_str_digits(100000)


        l =
        list(str(fact))
        ans = 0
        for i in range(len(l)-1, -1, -
            1):if l[i] == '0':
                ans += 1


            if l[i] !=
                '0':break


        return ans
```

OUTPUT:

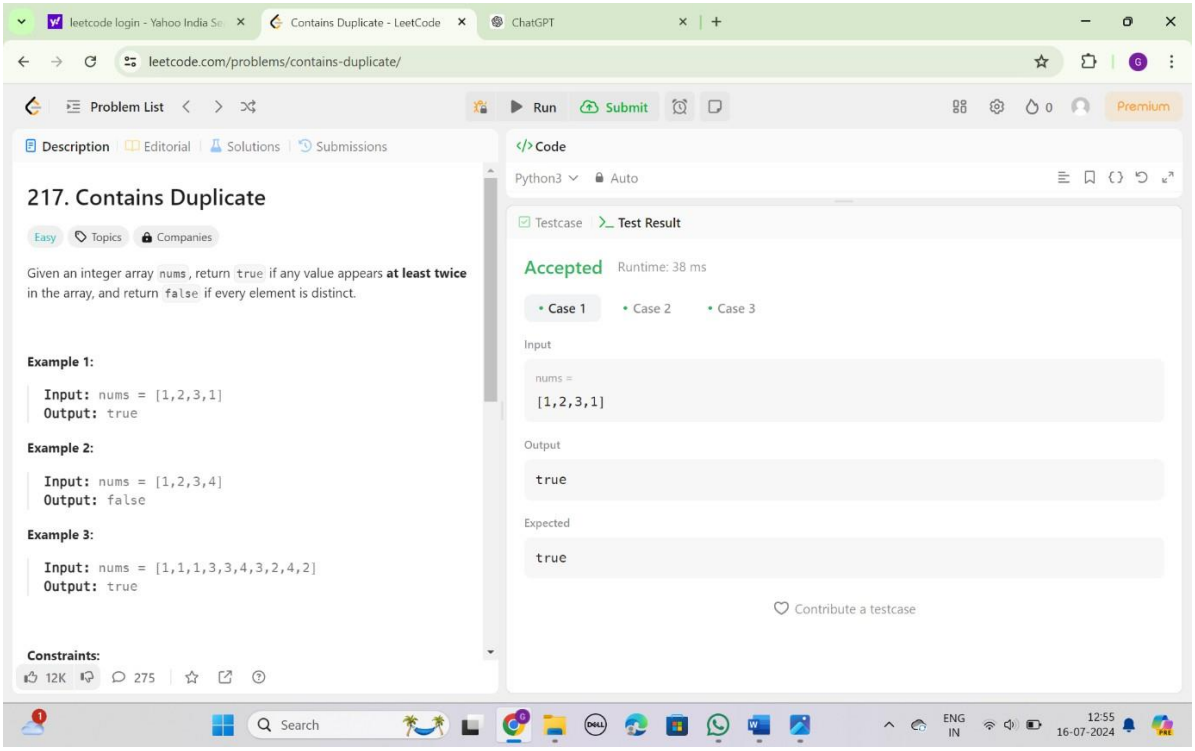


6. Time and Trade Off

6.1 Contains DuplicateCode:

```
class Solution:
    def containsDuplicate(self, nums: List[int]) ->
        bool:n = len(nums)
        for i in range(n - 1):
            for j in range(i + 1, n):
                if nums[i] ==
                    nums[j]:return
                    True
        return False
```

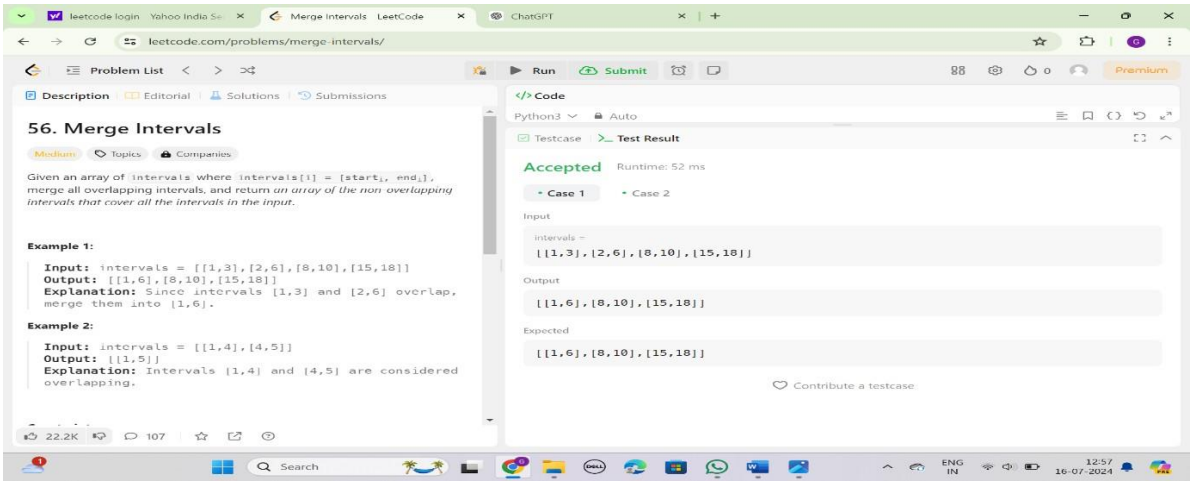
OUTPUT:



6.2 merge intervalsCode:

```
    class Solution:
  def merge(self, intervals: List[List[int]]) -> List[List[int]]:
    merged = []
    intervals.sort(key=lambda x: x[0])

    prev = intervals[0]

    for interval in intervals[1:]:
      if interval[0] <= prev[1]:
        prev[1] = max(prev[1], interval[1])
      else:
        merged.append(prev)
        prev = interval

    merged.append(prev)

    return merged
```

OUTPUT:



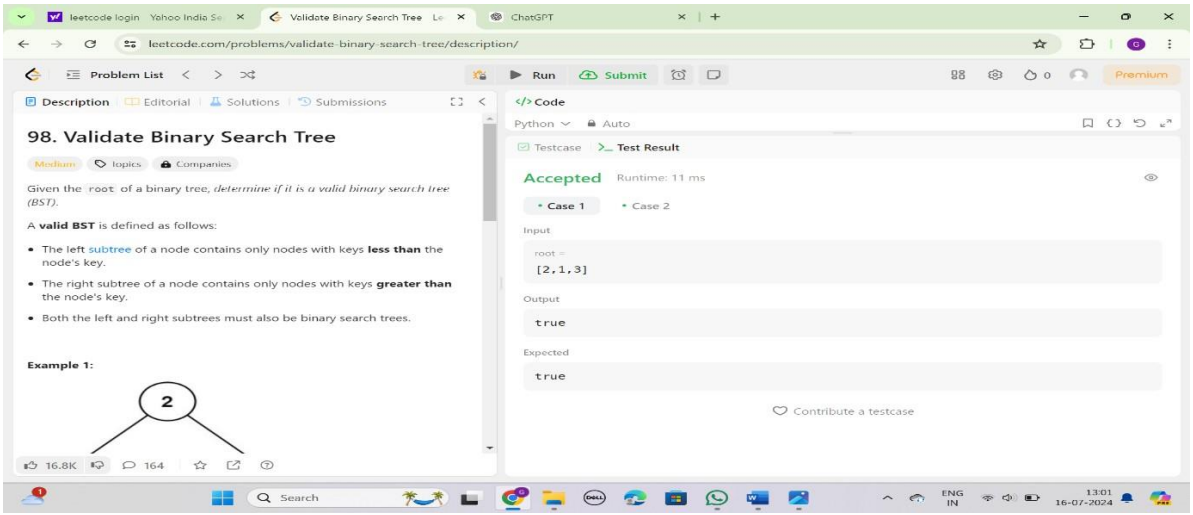
7.decrease and conquer

7.1 Validate binary search

tressCode:

```
class Solution(object):
    def helper(self, root, prev, result):
        if root is None: return
        self.helper(root.left, prev, result)
        if prev[0] and root.val <=
            prev[0].val:result[0] = False
            return
        prev[0] = root
        self.helper(root.right, prev, result)

    def isValidBST(self,
        root):prev = [None]
        result = [True]
        self.helper(root, prev, result)
        return result[0]
```

OUTPUT:

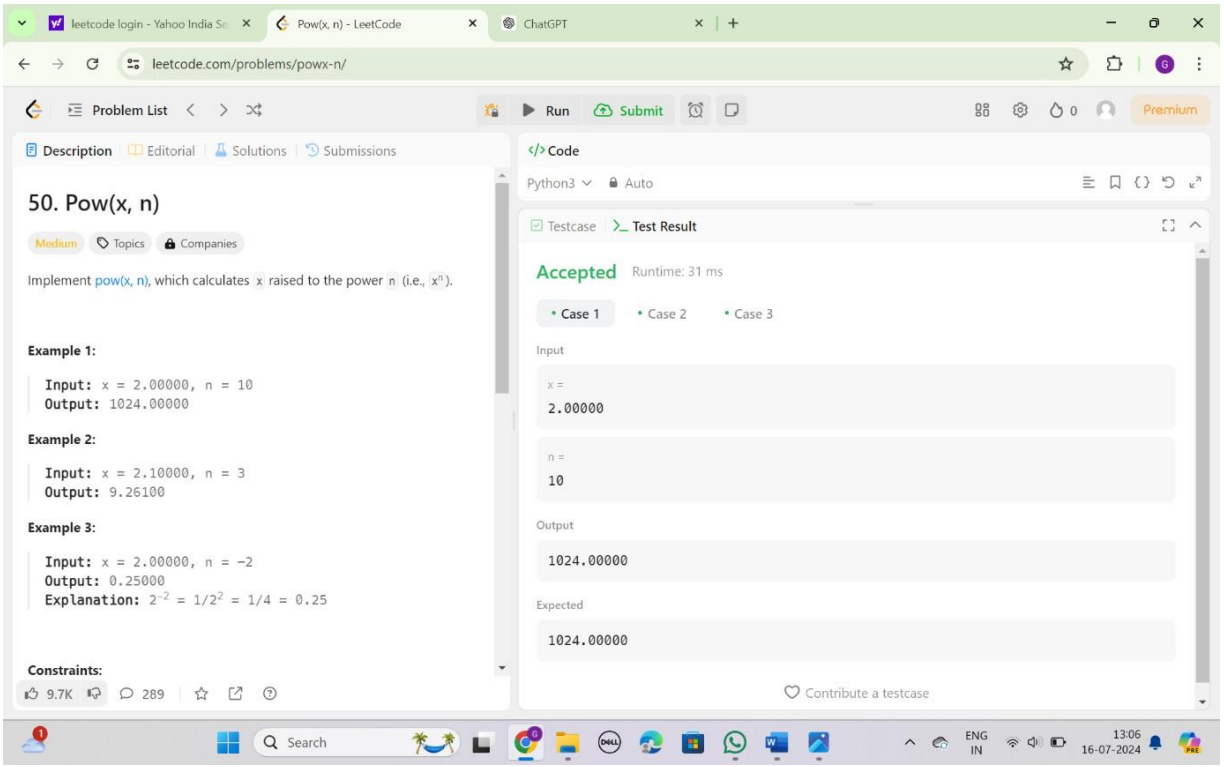


7.2 pow(x,n)

Code:

```
class Solution:
    def myPow(self, x: float, n: int) -> float:


        def calc_power(x, n):
            if x == 0:
                return 0
            if n == 0:
                return 1
            res = calc_power(x, n // 2)
            res = res * res

            if n % 2 == 1:
                return res * x
            return res
        ans = calc_power(x, abs(n))
        if n >= 0:
            return ans
        return 1 / ans
```

OUTPUT:



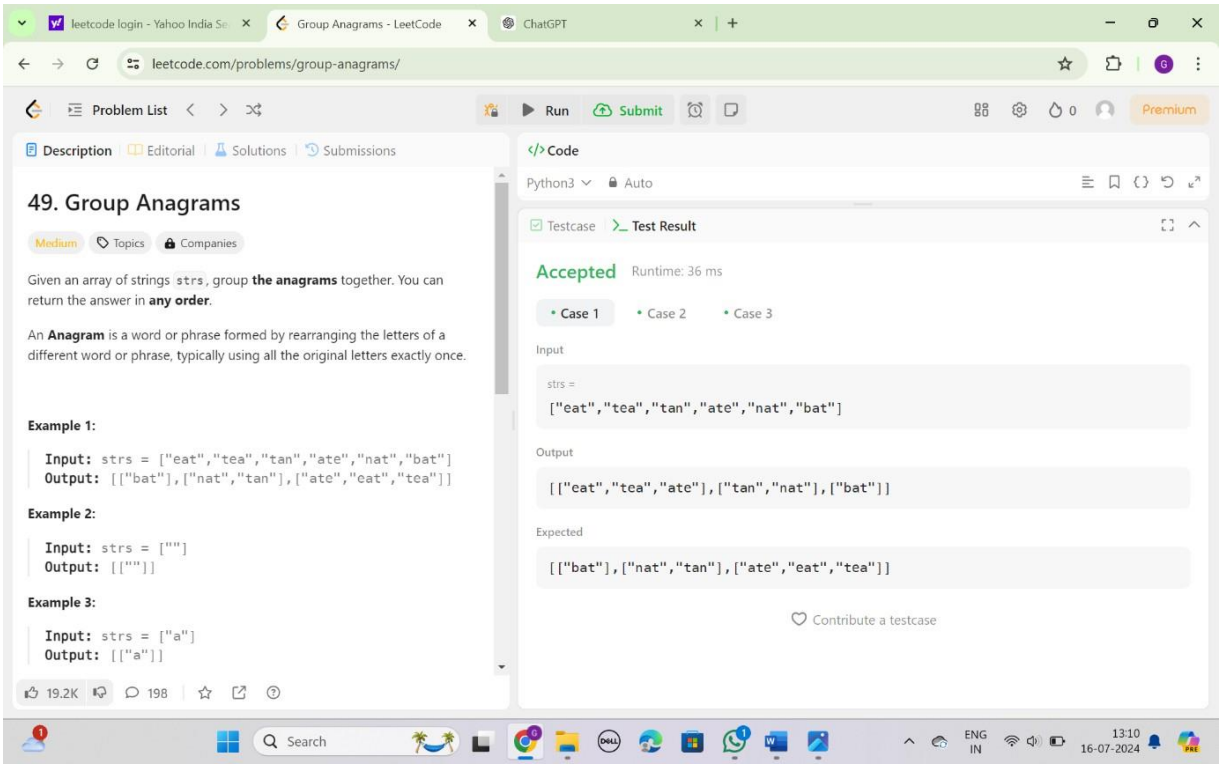
8.Transform Conquare

8.1 group

anagramsCode:

```
class Solution:
    def groupAnagrams(self, strs):
        anagram_map = defaultdict(list)

        for word in strs:
            sorted_word = ''.join(sorted(word))
            anagram_map[sorted_word].append(word)

        return list(anagram_map.values())
```

OUTPUT:



8.2 Integer to

RomanCode:

```
class Solution:
    def intToRoman(self, num: int) -> str:
        Roman = ""
        storeIntRoman = [[1000, "M"], [900, "CM"], [500, "D"], [400, "CD"], [100, "C"], [90, "XC"], [50, "L"], [40, "XL"], [10, "X"], [9, "IX"], [5, "V"], [4, "IV"], [1, "I"]]
        for i in range(len(storeIntRoman)):
            while num >=
                storeIntRoman[i][0]:Roman +=
                storeIntRoman[i][1] num -=
                storeIntRoman[i][0]
        return Roman
```

OUTPU