Date : 10/01/2023

Name : K Prasad Achari

1. 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.

Code:

class Solution:

    def findPeakElement(self, nums: List[int]) -> int:

        l=0

        h=len(nums)-1

        while l<h:

            m=(l+h)//2

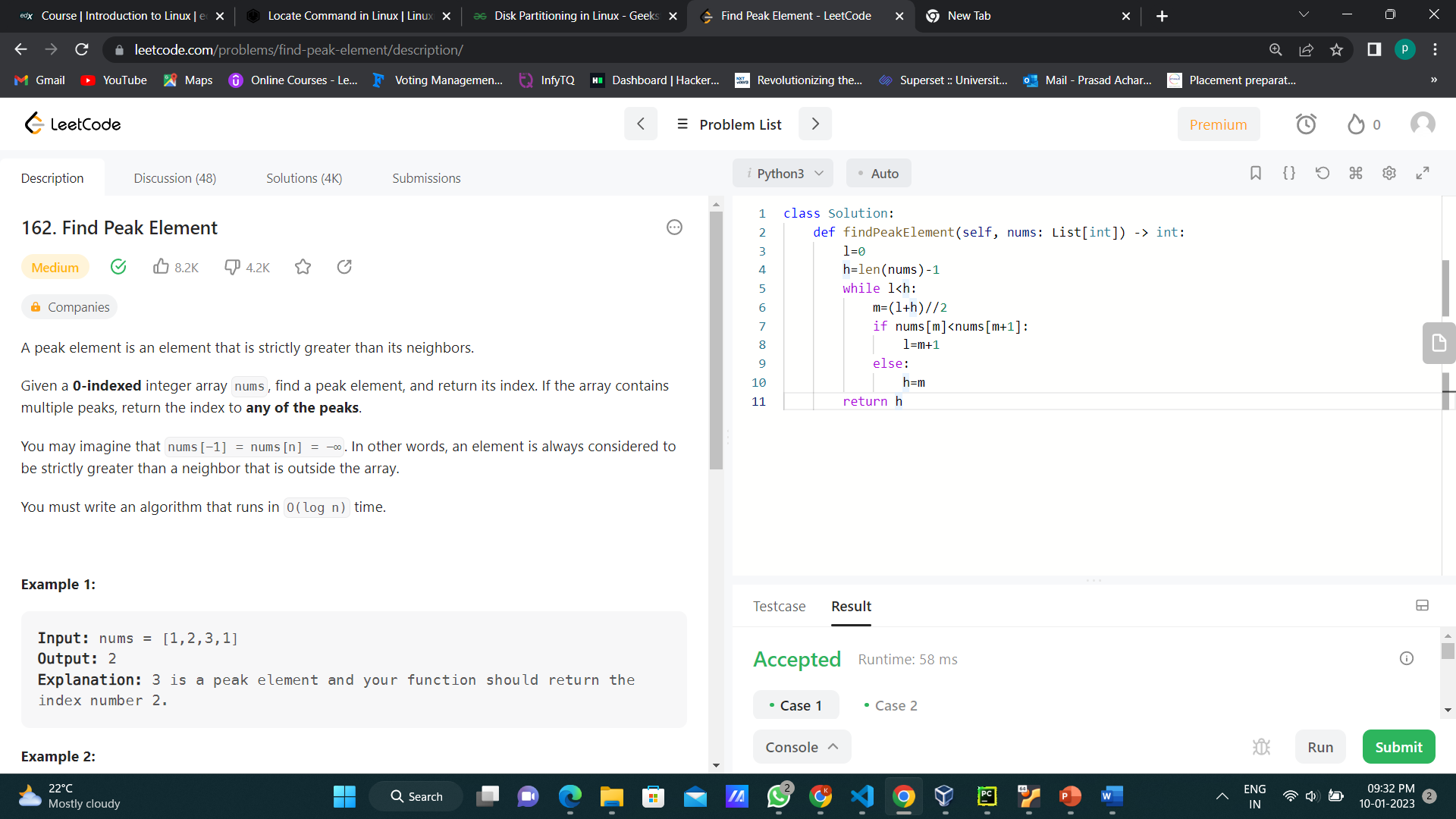
            if nums[m]<nums[m+1]:

                l=m+1

            else:

                h=m

        return h



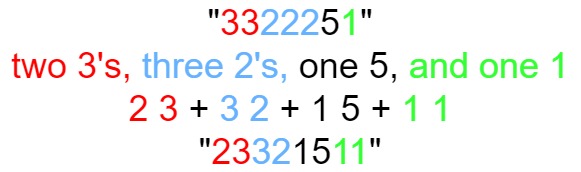
2. Count and Say

The **count-and-say** sequence is a sequence of digit strings defined by the recursive formula:

* countAndSay(1) = "1"
* countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string.

To determine how you "say" a digit string, split it into the **minimal** number of substrings such that each substring contains exactly **one** unique digit. Then for each substring, say the number of digits, then say the digit. Finally, concatenate every said digit.

For example, the saying and conversion for digit string "3322251":



Given a positive integer n, return *the*nth*term of the****count-and-say****sequence*.

**Example 1:**

**Input:** n = 1

**Output:** "1"

**Explanation:** This is the base case.

**Example 2:**

**Input:** n = 4

**Output:** "1211"

**Explanation:**

countAndSay(1) = "1"

countAndSay(2) = say "1" = one 1 = "11"

countAndSay(3) = say "11" = two 1's = "21"

countAndSay(4) = say "21" = one 2 + one 1 = "12" + "11" = "1211"

Code:

class Solution:

    def countAndSay(self, n: int) -> str:

        if n==1:

            return '1'

        s='1'

        for i in range(2,n+1):

            k=''

            o=1

            d=s[0]

            for j in s[1:]:

                if j==d:

                    o+=1

                else:

                    k+=str(o)+d

                    d=j

                    o=1

            k+=str(o)+d

            s=k

        return s

