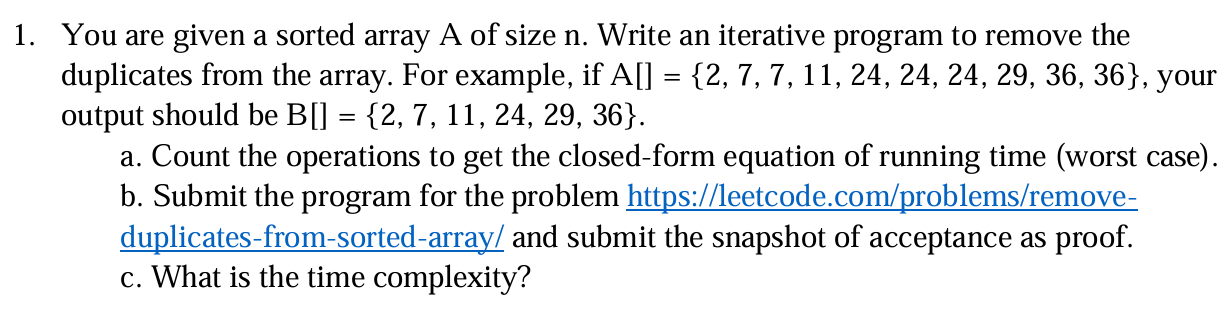
**22AIE212-Design and Analysis of Algorithms**

**Lab Sheet 2**

**Iterative programs-Part 2**

Name : Anuvind M P

Roll no: AM.EN.U4AIE22010

****

**CODE :**

def removeDuplicates(nums):

        a = []

        for i in range(len(nums)):

            if nums[i] != nums[i-1]:

                a.append(nums[i])

        nums = a

        return nums

A = [2, 7, 7, 11, 24, 24, 24, 29, 36, 36]

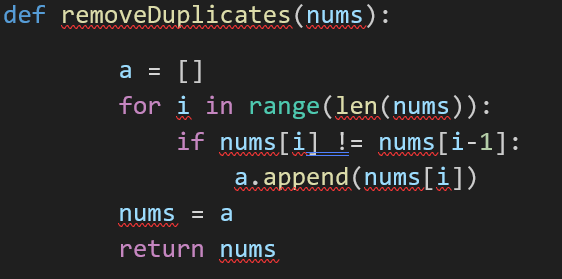
print(removeDuplicates(A))

**OUTPUT :**



**a.**

T(n)



1

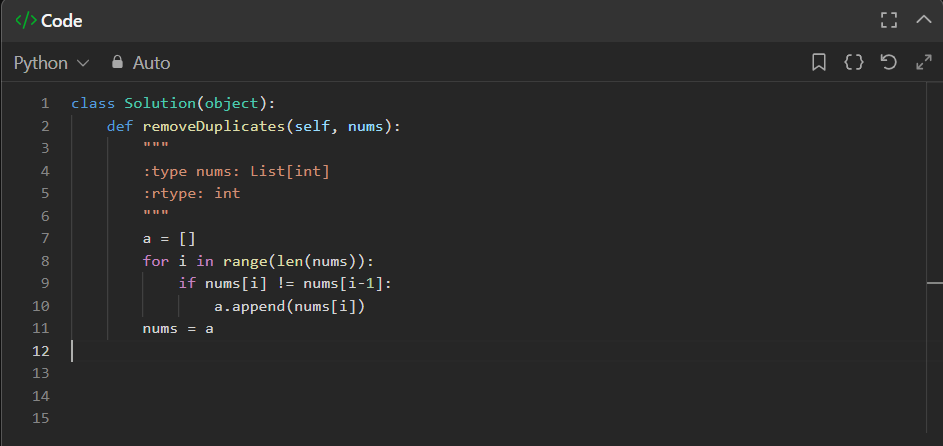
n

1

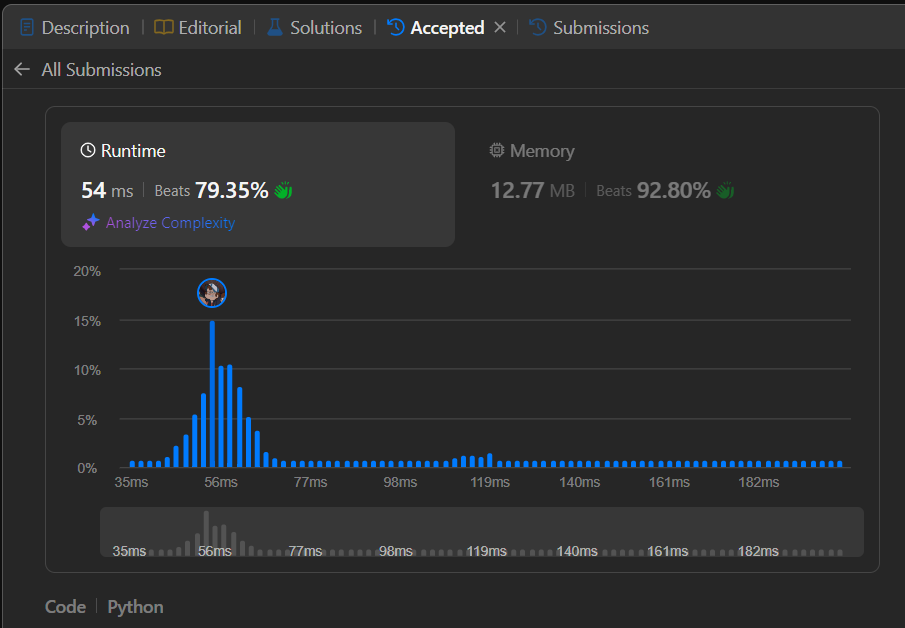
* T(n)=1+ n +1 = n+3
* T(n) = O(n)

**b.**

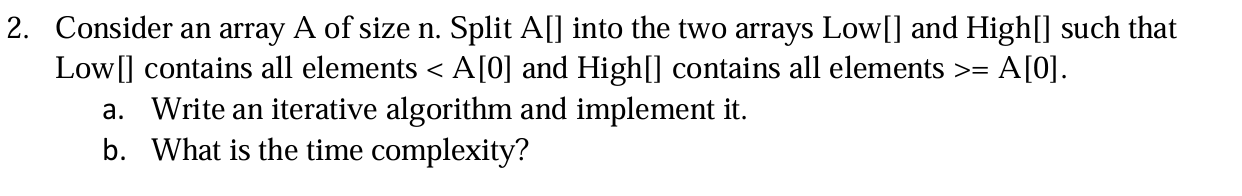
**Leetcode :**



**Leetcode :**



**c.** **Time Complexity = O(n)**

****

**CODE :**

def splitArray(nums):

    Low, High, chk= [], [], nums[0]

    for i in nums:

        if i < chk:

            Low.append(i)

        else:

            High.append(i)

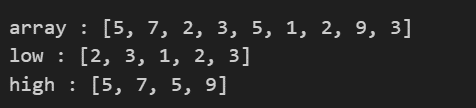
    return Low,High

nums = [5,7,2,3,5,1,2,9,3]

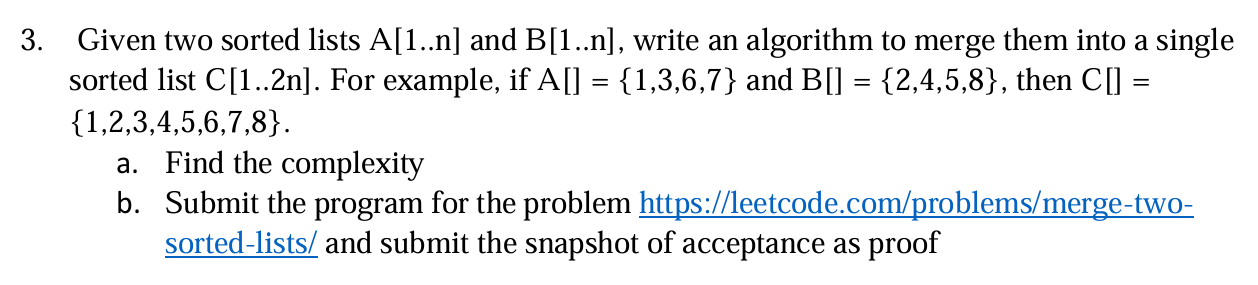
a,b = splitArray(nums)

print(f"low : {a} \nhigh : {b}")

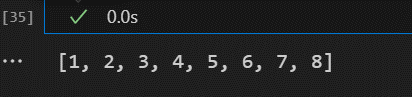
**OUTPUT :**



**Time complexity : O(n)**

****

**OUTPUT :**



**CODE :**

def merge(A, B):

    res = []

    i,j = 0, 0

    while i < len(A) and j < len(B):

        if A[i] < B[j]:

            res.append(A[i])

            i+=1

        else:

            res.append(B[j])

            j+=1

    if i<len(A):

        res+=A[i:]

    if j<len(B):

        res+=B[j:]

    return res

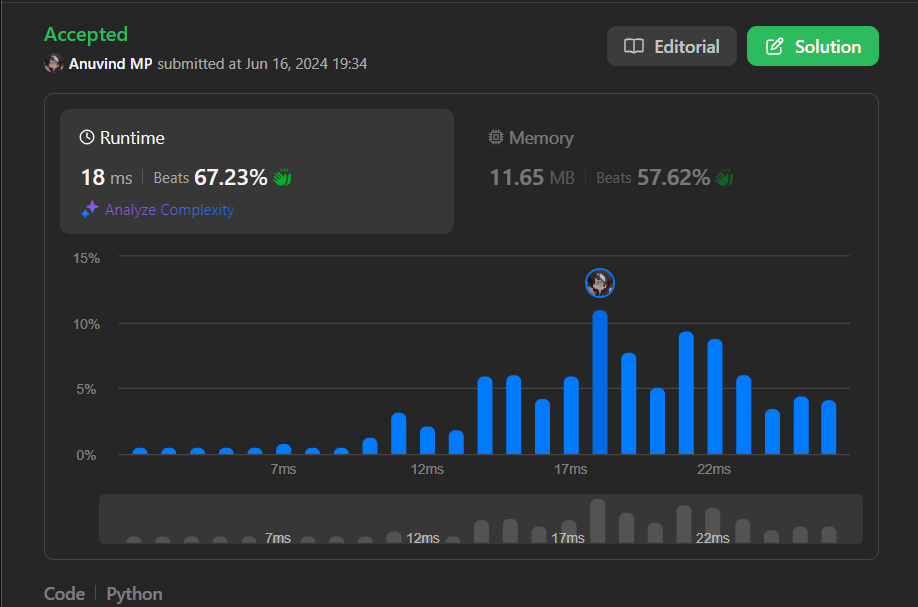
A = [1,3,6,7]

B = [2,4,5,8]

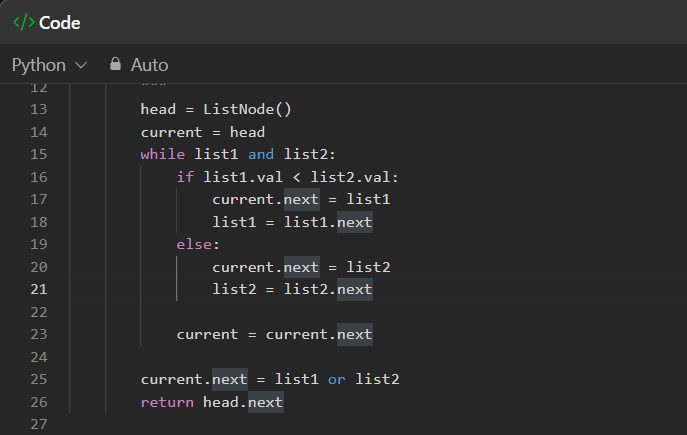
print(merge(A,B))

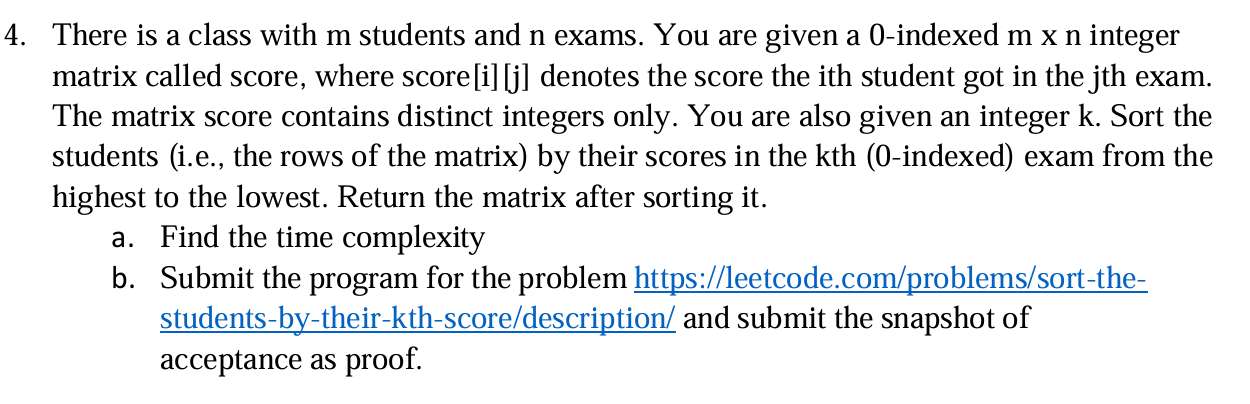
* **Time complexity : O(n)**
* **n = len(A) + len(B)**

**Leetcode :**



**Leetcode :**



****

**CODE :**

def sortTheStudents(score, k):

        output = []

        x = [i[k] for i in score]

        graph = {}

        j = 0

        for i in x:

            graph[i] = j

            j+=1

        x.sort(reverse = True)

        for i in x:

            key = graph[i]

            output.append(score[key])

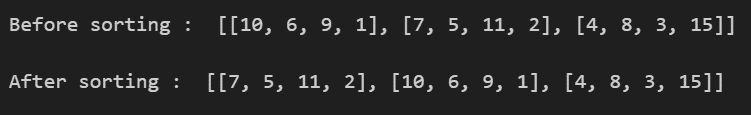
        return output

score = [[10,6,9,1],[7,5,11,2],[4,8,3,15]]

print("Before sorting : ",score)

print("\nAfter sorting : ",sortTheStudents(score,2))

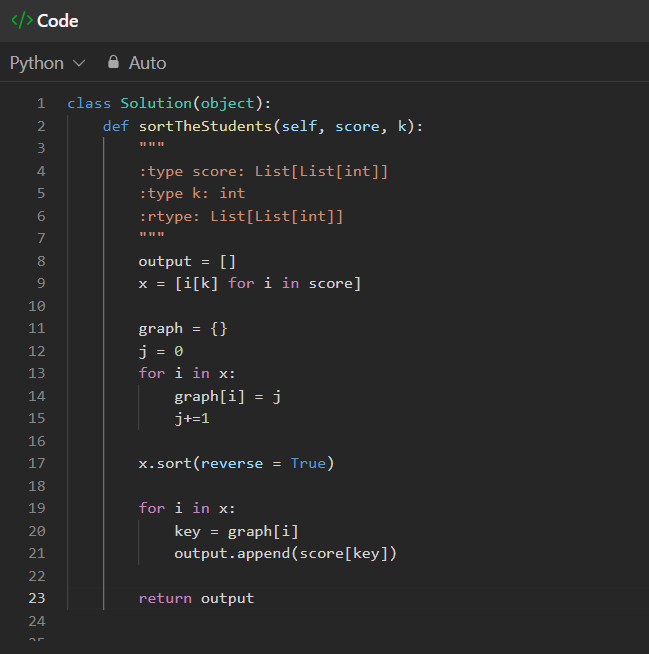
**OUTPUT :**



**Time Complexity :**

* **T(n) = n + n + nlogn + n = O(nlogn)**
* *Sorting takes nlogn time*

**Leetcode :**



**Leetcode :**

