# Lab Sheet 1

**Name: Vinayak V Thayil**

**Roll No: AM.EN.U4CSE21161**

1. Write a program to find the smallest element of an unsorted array of size N.

Don’t use any built-in functions/ methods like min() supported by certain programming languages.

arr=[]

n=int(input("Enter the size of the array: "))

print("Enter the elements of the array: ")

for i in range(n):

    arr.append(int(input()))

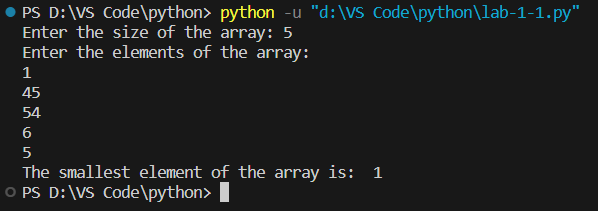
min=arr[0]

for i in range(n):

    if arr[i]<min:

        min=arr[i]

print("The smallest element of the array is: ",min)



* 1. How many times does your loop execute?

**n-1 times**

* 1. As the elements in the array changes (the size of the array remains same), will there be any change in the number of times the loop executes? What is the minimum and maximum number of times the loop executes?

**No Change and the minimum and maximum number is n-1**

* 1. What is the time complexity of your program?

**O(n)**

1. Write a program to find the smallest and the largest element in a sorted array.

arr=[]

n=int(input("Enter the number of elements in array:"))

for i in range(0,n):

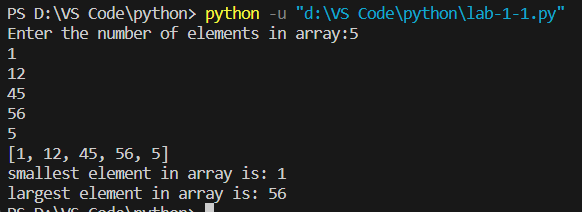
    ele=int(input())

    arr.append(ele)

print(arr)

print("smallest element in array is:",min(arr))

print("largest element in array is:",max(arr))



* 1. need any loops in this program?

**No**

* 1. What is the time complexity of your program?

**O(n)**

1. Write a program to search an element ‘k’ in an unsorted array of size N.

def search(arr, N, k):

    for i in range(N):

        if arr[i] == k:

            return 1

    return -1

arr = list(map(int, input("Enter the elements: ").split()))

k = int(input("Enter the element to search: "))

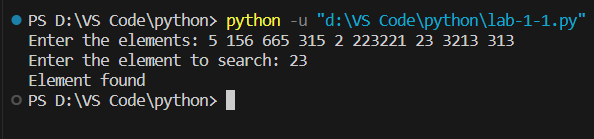
N = len(arr)

if search(arr, N, k) == 1:

    print("Element found")

else:

    print("Element not found")



* 1. As the elements in the array changes (the size of the array remains same), will there be any change in the number of times the loop executes? What is the minimum and maximum number of times the loop executes?

**Yes.**

**Maximum🡪O(n)**

**Minimum🡪1(1)**

* 1. What is the time complexity of your program?

**O(n)**

1. We need to search an element ‘k’ in a sorted array of size N.
   1. Will your program for Qn. No 3 work for this case?

**Yes**

* 1. Is the program for Qn. No 3 the most efficient one for this? (Hint: There exists a Binary search algorithm)

**No**

* 1. Write an iterative program to implement Binary search.

arr=[]

n=int(input("Enter the size :"))

for i in range(0,n):

    ele=int(input())

    arr.append(ele)

print(arr)

k=int(input("Enter the element to be searched:"))

low=0

high=n-1

mid=0

while(low<=high):

    mid=(low+high)//2

    if(arr[mid]<k):

        low=mid+1

    elif(arr[mid]>k):

        high=mid-1

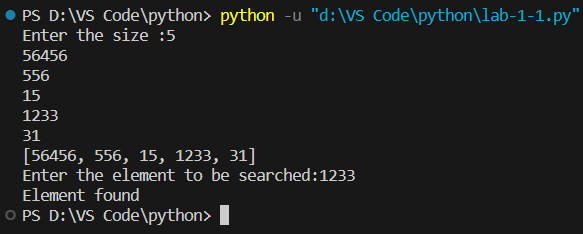
    else:

        print("Element found at index:")

        break

if(low>high):

    print("Element not found")



* 1. As the elements in the array changes (the size of the array remains same), will there be any change in the number of times the loop executes? What is the minimum and maximum number of times the loop executes?

**Yes.**

**Minimum🡪O(1)**

**Maximum🡪1(log n)**

* 1. What is the time complexity of your program?

**O(log n)**

1. Write an efficient program to find an element in an array which neither the smallest nor the largest. (Hint: you can do this without a loop.)

arr = []

for i in range(5):

    num = int(input("Enter a number: "))

    arr.append(num)

smallest = min(arr)

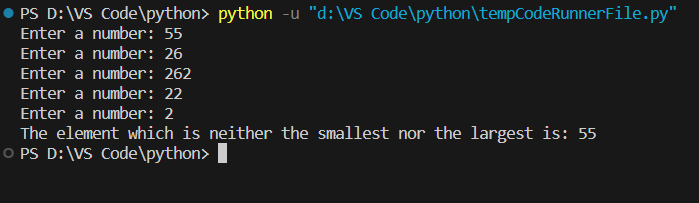
largest = max(arr)

arr.remove(smallest)

arr.remove(largest)

remaining = arr[0]

print("The element which is neither the smallest nor the largest is:", remaining)



* 1. What is the time complexity of your program?

**O(1)**

1. Write an efficient program to check if a given number is prime or not.

def is\_prime(n):

    if n <= 1:

        return False

    for i in range(2, n):

        if n % i == 0:

           return False

    return True

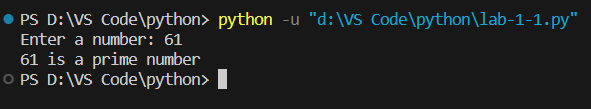
n = int(input("Enter a number: "))

if is\_prime(n):

    print(n, "is a prime number")

else:

    print(n, "is not a prime number")



* 1. What is the time complexity of the algorithm?

**O(sqrt(n))**

* 1. Show that the problem can be solved in root(n) time.

def is\_prime(n):

    if n <= 1:

        return False

    for i in range(2, n):

        if n % i == 0:

           return False

    return True

n = int(input("Enter a number: "))

if is\_prime(n):

    print(n, "is a prime number")

else:

    print(n, "is not a prime number")

1. Write an efficient program to find the GCD (also called HCF) of two given numbers.

def gcd(a, b):

    if b == 0:

        return a

    else:

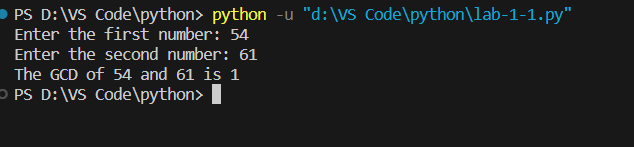
        return gcd(b, a % b)

a = int(input("Enter the first number: "))

b = int(input("Enter the second number: "))

result = gcd(a, b)

print("The GCD of", a, "and", b, "is", result)



* 1. What is the time complexity of the algorithm?

**O(log(min(a,b)))**

* 1. Find an input that requires maximum number of iterations to solve.

**The input that requires the maximum number of iterations to solve using the Euclidean algorithm is when the two input numbers are consecutive Fibonacci numbers.**

1. You are given a sorted array A of size n. Write an iterative program to remove the duplicates from the array. For example, if A[] = {2, 7, 7, 11, 24, 24, 24, 29, 36, 36}, your output should be B[] = {2, 7, 11, 24, 29, 36}.
   1. Count the operations to get the closed-form equation of running time (worst case).

class Solution(object):

     def removeDuplicates(self, nums):

        if len(nums) == 0:

            return 0

        i = 0

        for j in range(1,len(nums)):🡪n-1

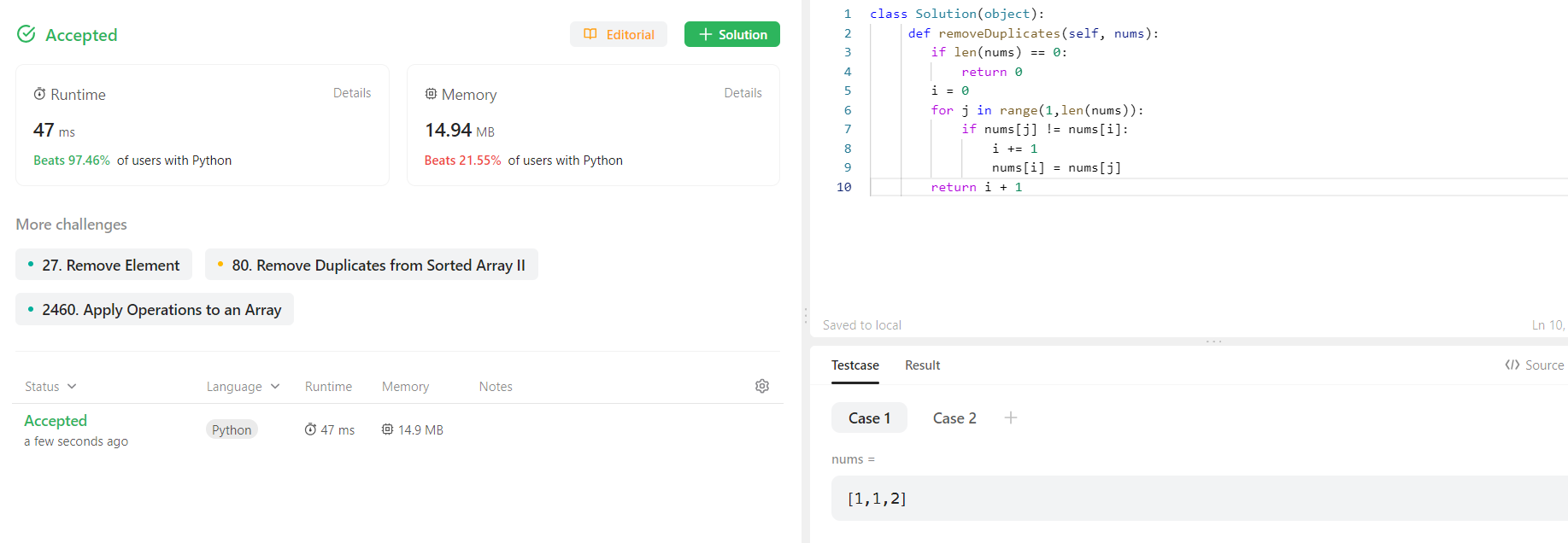
            if nums[j] != nums[i]:🡪n-1

                i += 1

                nums[i] = nums[j]

        return i + 1

* 1. Submit the program for the problem https://leetcode.com/problems/remove- duplicates-from-sorted-array/ and submit the snapshot of acceptance as proof.



* 1. What is the time complexity?

**O(n)**

1. Consider an array A of size n. Split A[] into the two arrays Low[] and High[] such that Low[] contains all elements < A[0] and High[] contains all elements >= A[0].
   1. Write an iterative algorithm and implement it.

def split\_array(A):

    Low = []

    High = []

    pivot = A[0]

    for x in A[1:]:

        if x < pivot:

            Low.append(x)

        else:

            High.append(x)

    return Low, High

* 1. What is the time complexity?

**O(nlogn)**

1. Given two sorted lists A[1..n] and B[1..n], write an algorithm to merge them into a single sorted list C[1..2n]. For example, if A[] = {1,3,6,7} and B[] = {2,4,5,8}, then C[] =

{1,2,3,4,5,6,7,8}.

def merge\_sorted\_lists(A, B):

    i = 0

    j = 0

    n = len(A)

    m = len(B)

    C = []

    while i < n and j < m:

        if A[i] <= B[j]:

            C.append(A[i])

            i += 1

        else:

            C.append(B[j])

            j += 1

    if i < n:

        C.extend(A[i:])

    if j < m:

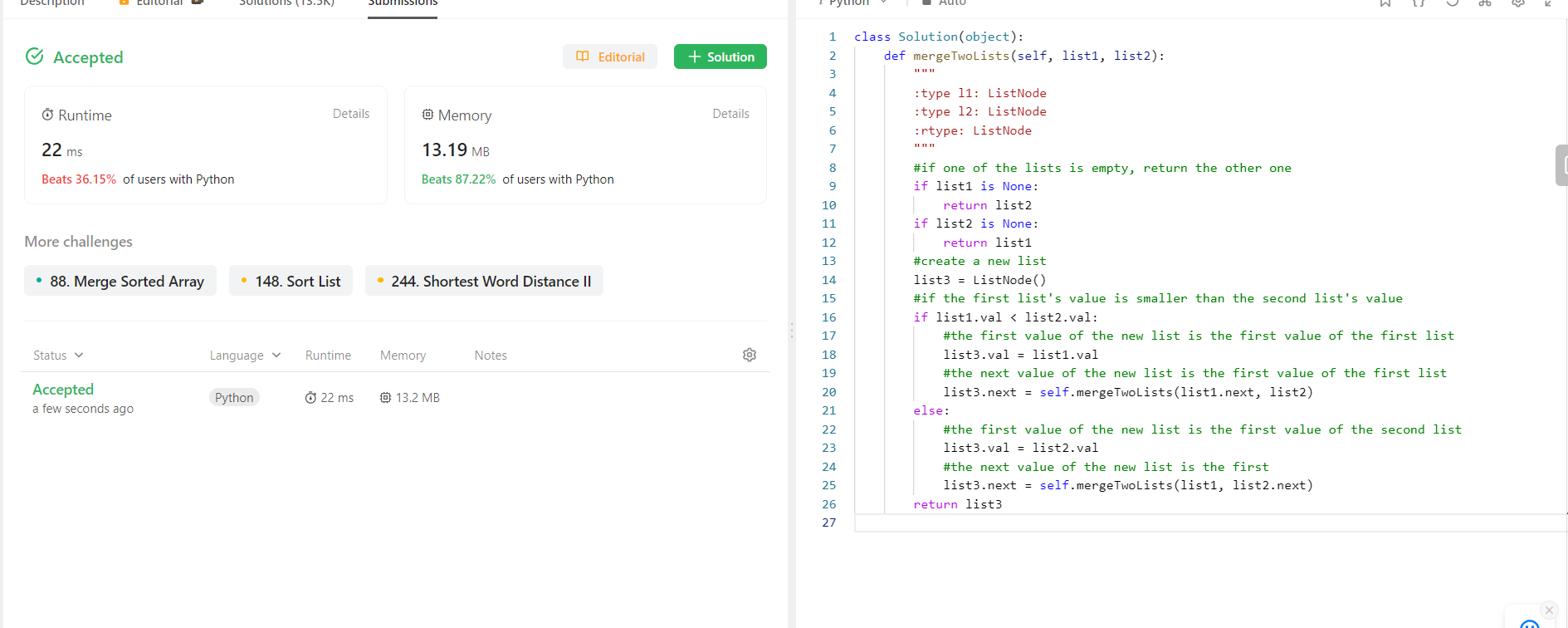
        C.extend(B[j:])

    return C

* 1. Find the complexity

**O(len(A) + len(B))**

* 1. Submit the program for the problem https://leetcode.com/problems/merge-two- sorted-lists/ and submit the snapshot of acceptance as proof



1. You are given an array coordinates, cord[i] = [x, y], where [x, y] represents the coordinate of a point. Check if these points make a straight line in the XY plane.

def checkStraightLine(coordinates):

    if len(coordinates) <= 2:

        return True

    x0, y0 = coordinates[0]

    x1, y1 = coordinates[1]

    for i in range(2, len(coordinates)):

        xi, yi = coordinates[i]

        if (yi - y0) \* (x1 - x0) != (y1 - y0) \* (xi - x0):

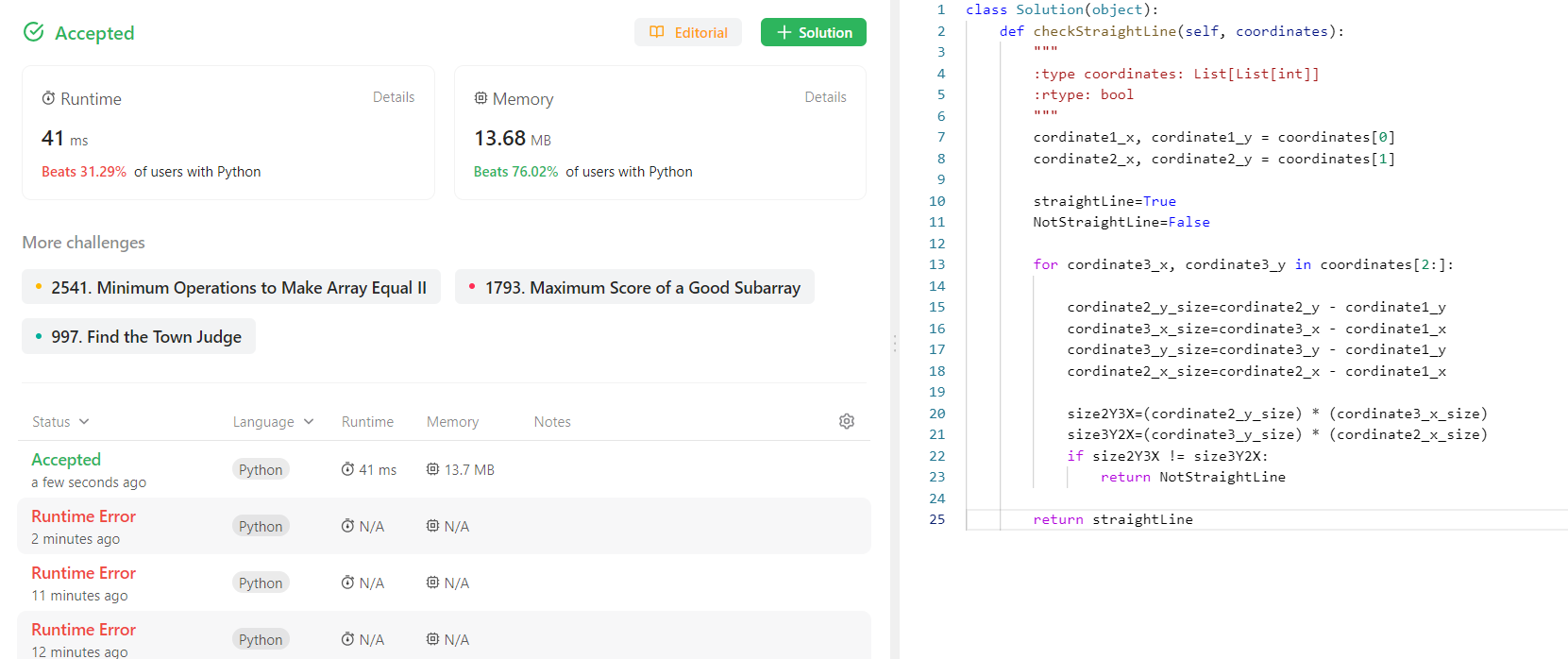
            return False

    return True

* 1. Find the time complexity of the algorithm.

**O (len(array))**

* 1. Submit the program for the problem https://leetcode.com/problems/check-if-it-is-a- straight-line/ and submit the snapshot of acceptance as proof.



1. There is a class with m students and n exams. You are given a 0-indexed m x n integer matrix called score, where score[i][j] denotes the score the ith student got in the jth exam. The matrix score contains distinct integers only. You are also given an integer k. Sort the students (i.e., the rows of the matrix) by their scores in the kth (0-indexed) exam from the highest to the lowest. Return the matrix after sorting it.

def sort\_scores(score, k):

    return sorted(score, key=lambda x: x[k], reverse=True)

* 1. Find the time complexity

**O(N log N)**

* 1. Submit the program for the problem https://leetcode.com/problems/sort-the- students-by-their-kth-score/ and submit the snapshot of acceptance as proof.

