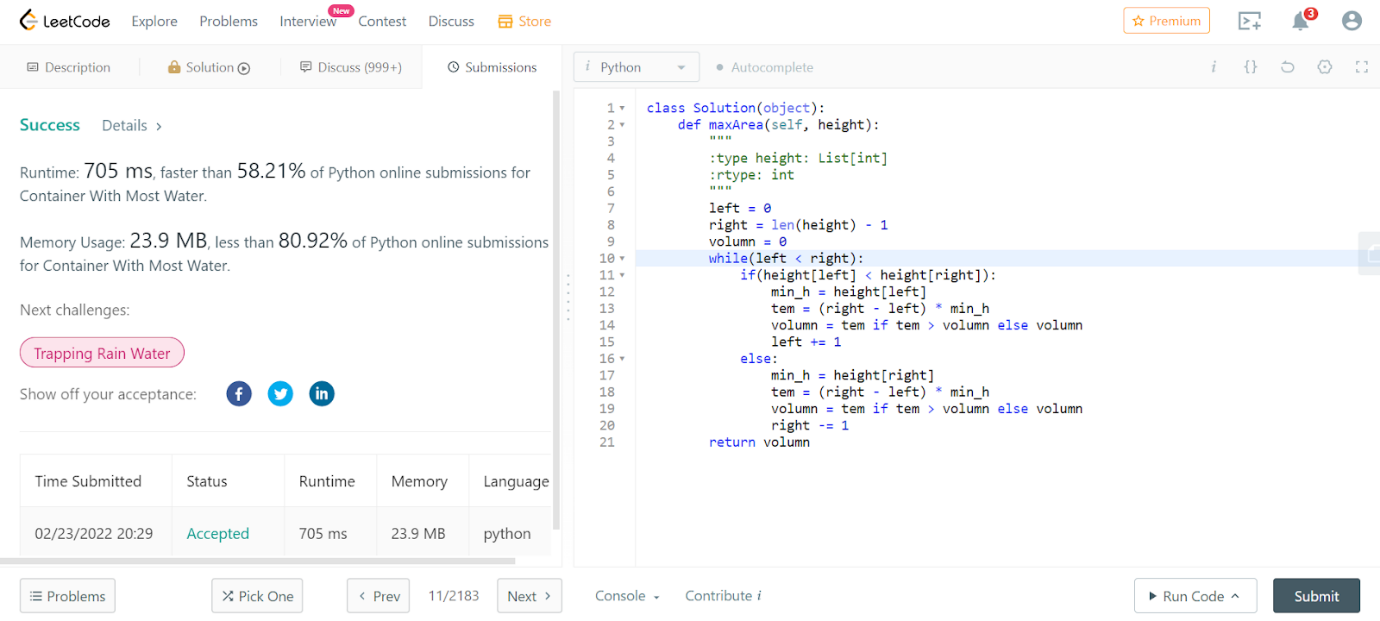
**11. Container with most water**



class Solution(object):

    def maxArea(self, height):

        """

        :type height: List[int]

        :rtype: int

        """

        left = 0

        right = len(height) - 1

        volumn = 0

        while(left < right):

            if(height[left] < height[right]):

                min\_h = height[left]

                tem = (right - left) \* min\_h

                volumn = tem if tem > volumn else volumn

                left += 1

            else:

                min\_h = height[right]

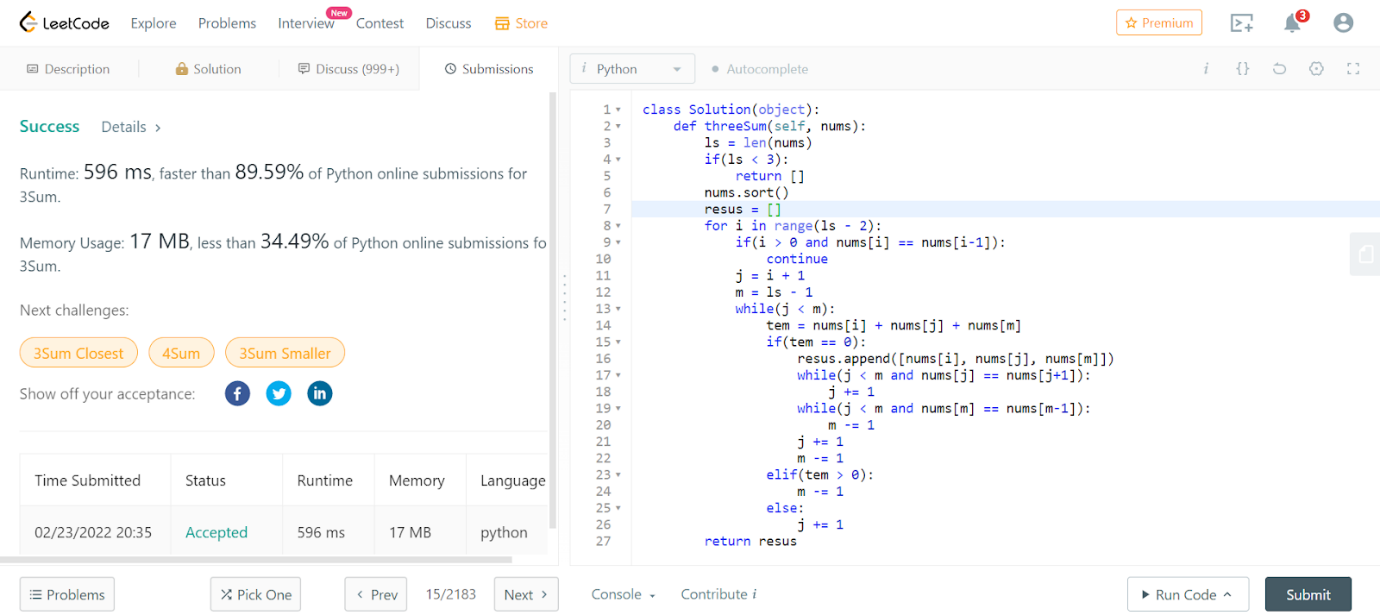
                tem = (right - left) \* min\_h

                volumn = tem if tem > volumn else volumn

                right -= 1

        return volumn

**15. 3Sum**



class Solution(object):

    def threeSum(self, nums):

        ls = len(nums)

        if(ls < 3):

            return []

        nums.sort()

        resus = []

        for i in range(ls - 2):

            if(i > 0 and nums[i] == nums[i-1]):

                continue

            j = i + 1

            m = ls - 1

            while(j < m):

                tem = nums[i] + nums[j] + nums[m]

                if(tem == 0):

                    resus.append([nums[i], nums[j], nums[m]])

                    while(j < m and nums[j] == nums[j+1]):

                        j += 1

                    while(j < m and nums[m] == nums[m-1]):

                        m -= 1

                    j += 1

                    m -= 1

                elif(tem > 0):

                    m -= 1

                else:

                    j += 1

        return resus

**16. 3Sum Closest**

Graphical user interface, text, application

Description automatically generated

class Solution(object):

    def threeSumClosest(self, nums, target):

        ls = len(nums)

        if(ls < 3):

            return []

        nums = sorted(nums)

        result = sum(nums[0:3])

        for i in range(ls - 2):

            j = i + 1

            m = ls - 1

            while(j < m):

                tem = nums[i] + nums[j] + nums[m]

                v = tem - target

                if(abs(v) < abs(result - target)):

                    result = tem

                if(v == 0):

                    return target

                elif(v > 0):

                    m -= 1

                else:

                    j += 1

        return result

**18. 4Sum**

Graphical user interface, application

Description automatically generated

class Solution(object):

    def fourSum(self, nums, target):

        """

        :type nums: List[int]

        :type target: int

        :rtype: List[List[int]]

        """

        num = sorted(nums)

        n = len(nums)

        m = collections.defaultdict(list)

        res = set()

        for i in xrange(n-1):

            for j in xrange(i+1,n):

                m[num[i]+num[j]].append((i,j))

        for i in xrange(n-1):

            for j in xrange(i+1,n):

                rest = target-num[i]-num[j]

                for pair in m[rest]:

                    if i>pair[1]:

                        res.add((num[pair[0]], num[pair[1]], num[i], num[j]))

        return list(res)