

## ASSIGNMENT 3

➤ Question 1:

**Implementation of Merge sort.**

**TC:  $O(n \log n)$**

➤ Solution:

- Source Code:

```
def merge(left, right):
    merged = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            merged.append(left[i])
            i += 1
        else:
            merged.append(right[j])
            j += 1
    while i < len(left):
        merged.append(left[i])
        i += 1
    while j < len(right):
        merged.append(right[j])
        j += 1
    return merged

def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
    left_half = merge_sort(arr[:mid])
    right_half = merge_sort(arr[mid:])
    return merge(left_half, right_half)

arr = [38, 27, 43, 3, 9, 82, 10]
```

```
sorted_arr = merge_sort(arr)
print("Sorted array:", sorted_arr)
```

- Output:



GANDHI UDAY



NEW

PYTHON ▼

RUN ▶



STDIN

Input for the program ( Optional )

Output:

Sorted array: [3, 9, 10, 27, 38, 43, 82]

➤ Question 2:

**Implementation of Max-Min by using Divide and Conquer principal**

**TC:  $O(n)$**




➤ Solution:


- Source code:

```
def find_max_min(arr, low, high):  
    if low == high:  
        return arr[low], arr[low]  
  
    elif high == low + 1:  
        if arr[low] > arr[high]:  
            return arr[low], arr[high]  
        else:  
            return arr[high], arr[low]  
  
    mid = (low + high) // 2  
    max1, min1 = find_max_min(arr, low, mid)  
    max2, min2 = find_max_min(arr, mid + 1, high)  
  
    overall_max = max(max1, max2)  
    overall_min = min(min1, min2)  
  
    return overall_max, overall_min  
  
arr = [3, 5, 1, 8, 9, 2, 7, 6]  
n = len(arr)  
maximum, minimum = find_max_min(arr, 0, n - 1)  
print(f"Maximum element: {maximum}")  
print(f"Minimum element: {minimum}")
```

- Output:

---

 GANDHI UDAY

 AI

NEW

PYTHON ▾

RUN ▶

⋮

STDIN

Input for the program ( Optional )

---

Output:

Maximum element: 9  
Minimum element: 1

➤ Question 3:

**Fractional Knapsack GeeksForGeeks Implementation of Fractional KnapSack TC:  $O(n \log n)$  (Problem Statement: The weight of  $N$  items and their corresponding values are given. We have to put these items in a knapsack of weight  $W$  such that the total value obtained is maximized.)**

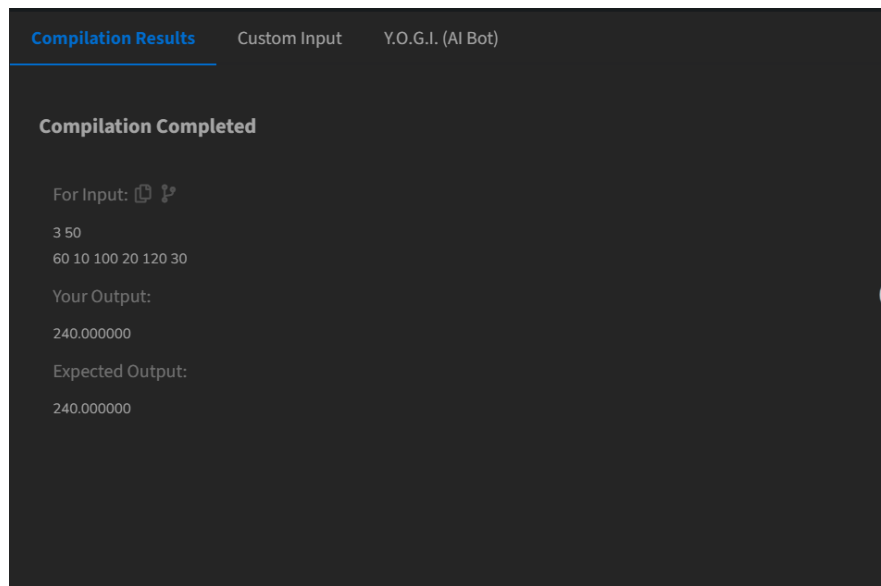
➤ Solution:

- Source code:

```
class Item:
    def __init__(self, val, w):
        self.value = val
        self.weight = w

class Solution:
    def fractionalknapsack(self, w, arr, n):
        prof = [arr[i].value / arr[i].weight for i in range(n)]
        items = [[prof[i], arr[i].value, arr[i].weight] for i in range(n)]
        items.sort(key=lambda x: x[0], reverse=True)
        profit = 0
        i = 0
        while w > 0 and i < n:
            if items[i][2] <= w:
                profit += items[i][1]
                w -= items[i][2]
            else:
                profit += items[i][0] * w
                w = 0
            i += 1
        return profit
```

- Output:



The screenshot shows a dark-themed interface with a tabbed header. The 'Compilation Results' tab is selected. Below the header, the text 'Compilation Completed' is displayed. Underneath, it shows 'For Input:' followed by a file icon and a text icon. The input values are '3 50' and '60 10 100 20 120 30'. The 'Your Output:' is '240.000000' and the 'Expected Output:' is also '240.000000'.

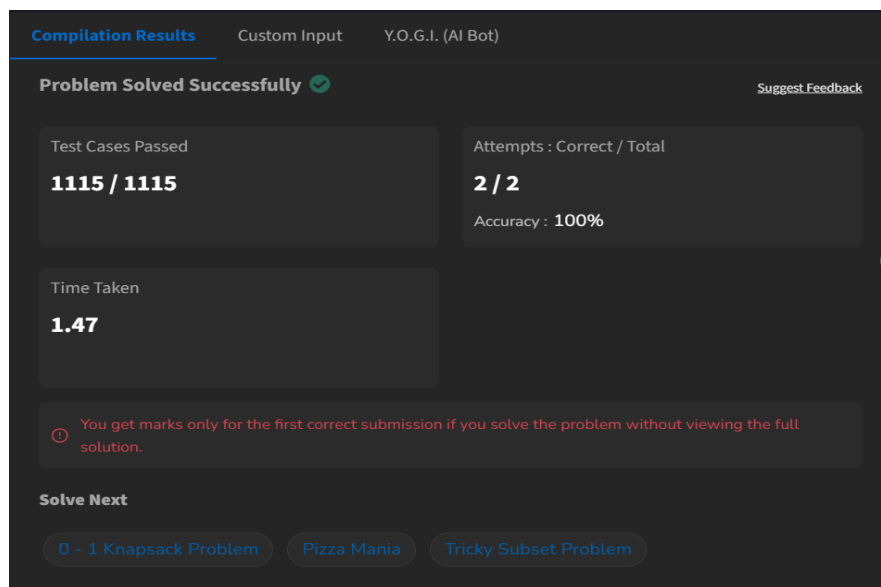
```
Compilation Results Custom Input Y.O.G.I. (AI Bot)

Compilation Completed

For Input: [file icon] [text icon]
3 50
60 10 100 20 120 30

Your Output:
240.000000

Expected Output:
240.000000
```



The screenshot shows a dark-themed interface with a tabbed header. The 'Compilation Results' tab is selected. Below the header, the text 'Problem Solved Successfully' is displayed with a green checkmark icon. To the right, there is a 'Suggest Feedback' link. Below this, there are two main sections: 'Test Cases Passed' showing '1115 / 1115' and 'Attempts : Correct / Total' showing '2 / 2' with 'Accuracy : 100%'. Below these, there is a 'Time Taken' section showing '1.47'. At the bottom, there is a message: 'You get marks only for the first correct submission if you solve the problem without viewing the full solution.' and a 'Solve Next' section with three buttons: '0 - 1 Knapsack Problem', 'Pizza Mania', and 'Tricky Subset Problem'.

```
Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully [checkmark icon] Suggest Feedback

Test Cases Passed 1115 / 1115 Attempts : Correct / Total 2 / 2 Accuracy : 100%

Time Taken 1.47

[info icon] You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Solve Next
0 - 1 Knapsack Problem Pizza Mania Tricky Subset Problem
```

➤ Question 4:  
**Implementation of Prim's Algorithm.**

➤ Solution:

- Source code:

```
import heapq

def prim(graph, start):
    mst = []
    visited = set()
    min_heap = [(0, start)]
    total_cost = 0

    while min_heap:
        cost, node = heapq.heappop(min_heap)
        if node in visited:
            continue
        visited.add(node)
        total_cost += cost
        mst.append((node, cost))

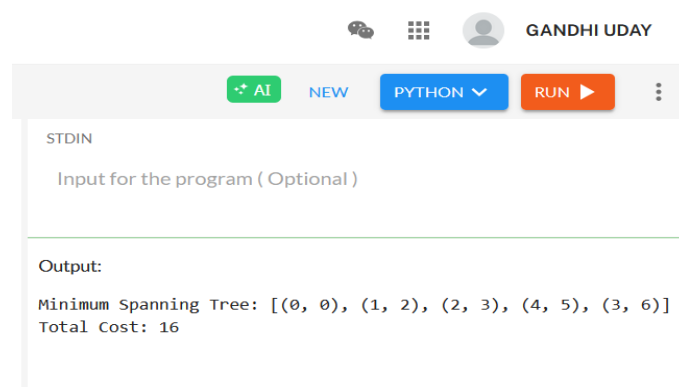
        for neighbor, weight in graph[node]:
            if neighbor not in visited:
                heapq.heappush(min_heap, (weight, neighbor))

    return mst, total_cost

graph = {
    0: [(1, 2), (3, 6)],
    1: [(0, 2), (2, 3), (3, 8), (4, 5)],
    2: [(1, 3), (4, 7)],
    3: [(0, 6), (1, 8)],
    4: [(1, 5), (2, 7)]
}

mst, total_cost = prim(graph, 0)
print("Minimum Spanning Tree:", mst)
print("Total Cost:", total_cost)
```

- Output:



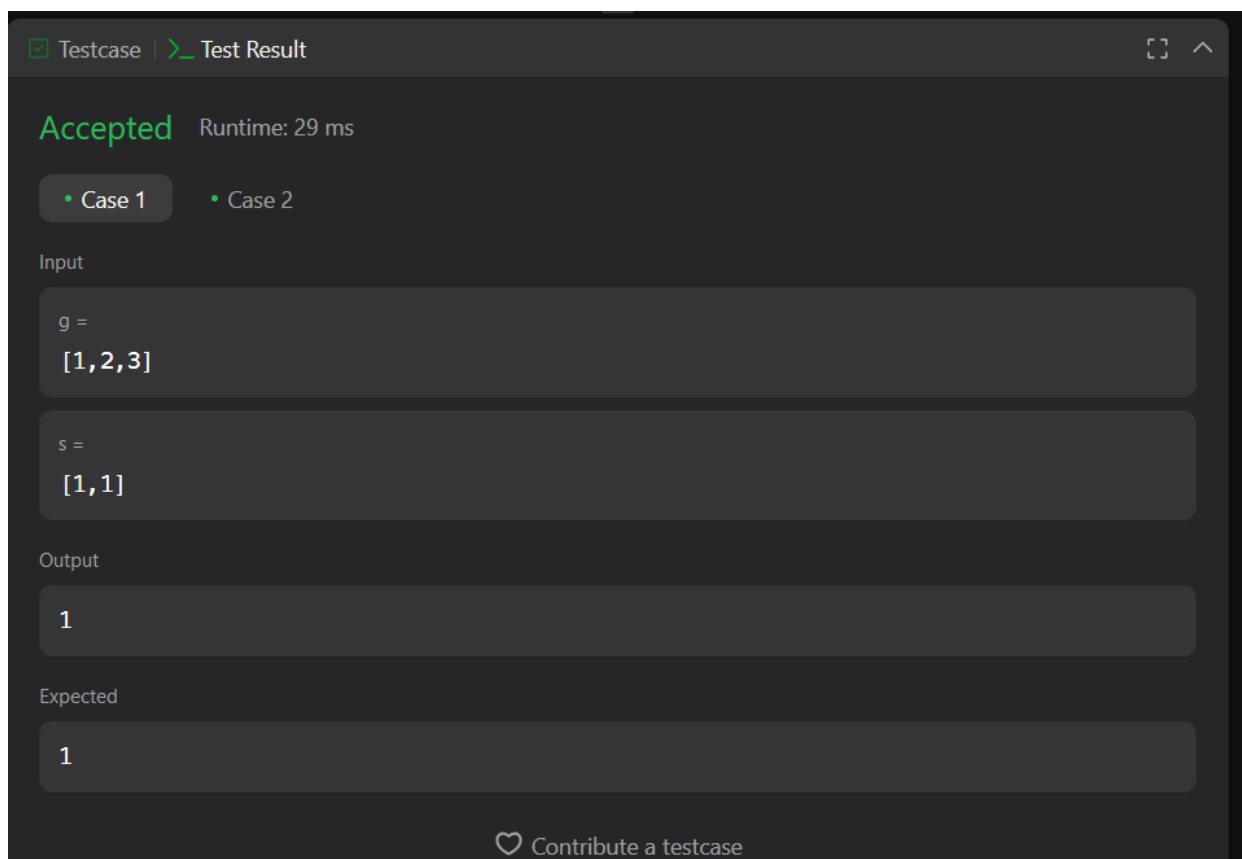
The screenshot shows a code editor interface with a header bar containing icons for chat, a grid, and a user profile labeled "GANDHI UDAY". Below the header, there are buttons for "AI", "NEW", "PYTHON" (with a dropdown arrow), and "RUN" (with a play icon). The main area is divided into two sections: "STDIN" and "Output". The "STDIN" section has a text input field with the placeholder "Input for the program ( Optional )". The "Output" section displays the result of the program execution: "Minimum Spanning Tree: [(0, 0), (1, 2), (2, 3), (4, 5), (3, 6)]" and "Total Cost: 16".

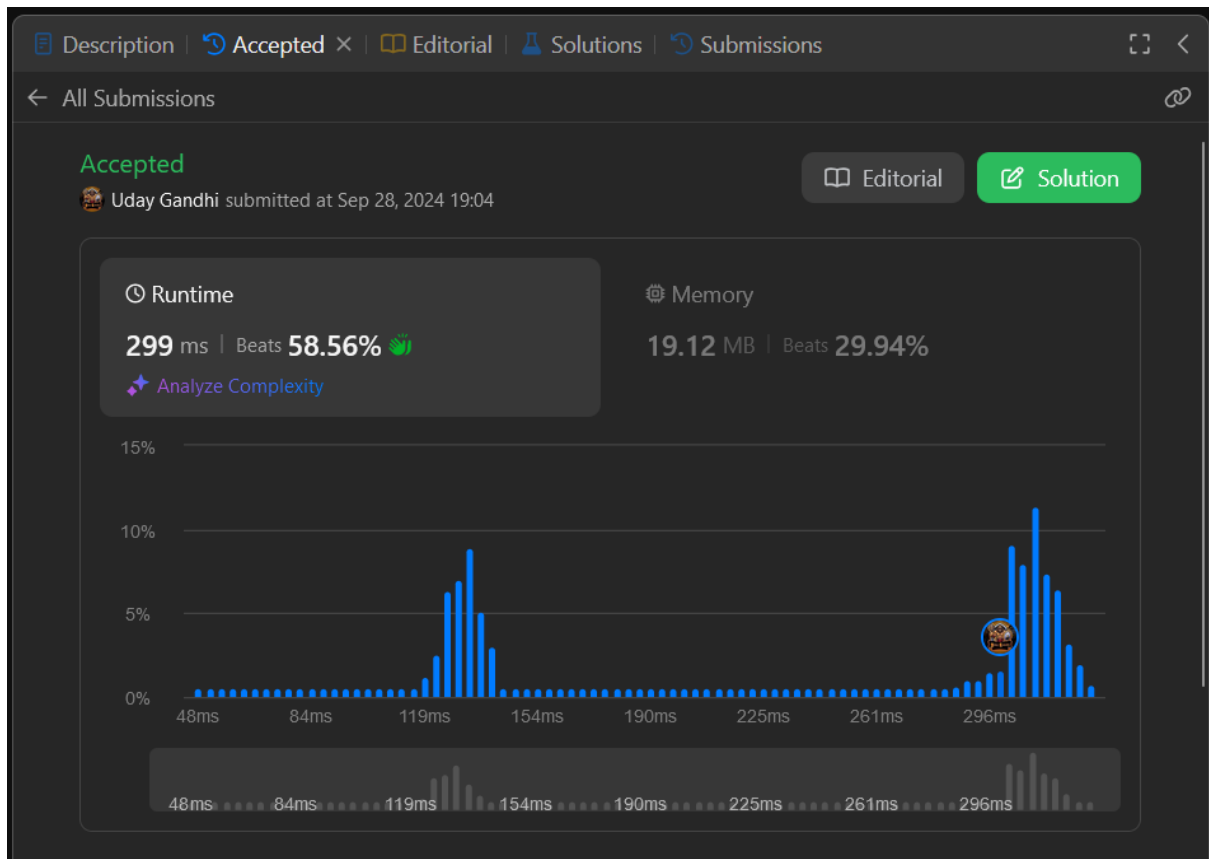
➤ Question 5:

**Assign Cookies.** (Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.) Leetcode problem number: 455

➤ Solution:• Source code:

```
def find_content_children(g, s):  
    g.sort()  
    s.sort()  
    i = j = 0  
    while i < len(g) and j < len(s):  
        if s[j] >= g[i]:  
            i += 1  
            j += 1  
    return i  
  
g = [1, 2, 3]  
s = [1, 1]  
result = find_content_children(g, s)  
print(result)
```

• Output:





➤ Question 6:**Maximum Units on a Truck. Leetcode problem number: 1710**➤ Solution:• Source Code:

class Solution:

def maximumUnits(self, boxTypes: List[List[int]], truckSize: int) -&gt; int:

boxTypes.sort(key=lambda x: x[1], reverse=True)

total\_units = 0

for box\_count, units in boxTypes:

if truckSize == 0:

break

if box\_count &lt;= truckSize:

total\_units += box\_count \* units

truckSize -= box\_count

else:

total\_units += truckSize \* units

truckSize = 0

return total\_units

• Solution:

Testcase | Test Result

**Accepted** Runtime: 33 ms

• Case 1 • Case 2

Input

boxTypes =  
[[1,3], [2,2], [3,1]]

truckSize =  
4

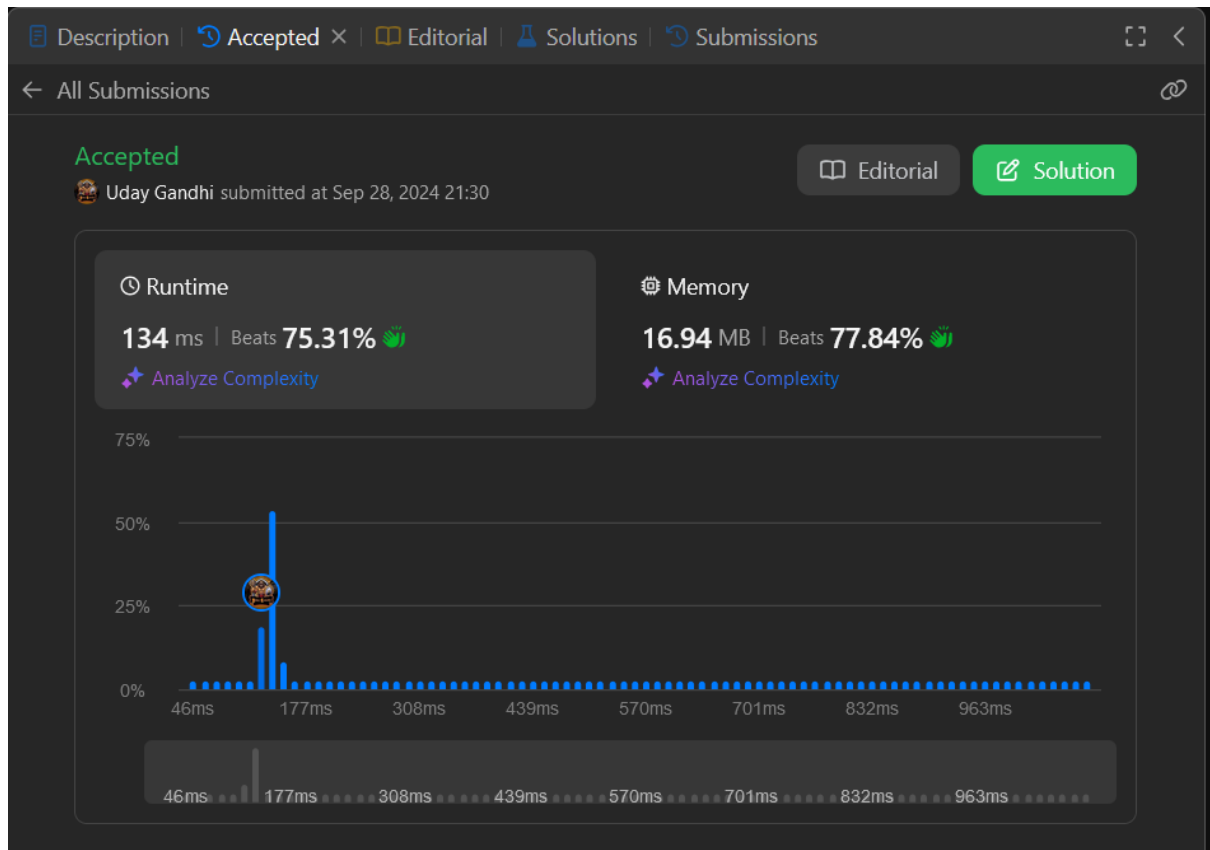
Output

8

Expected

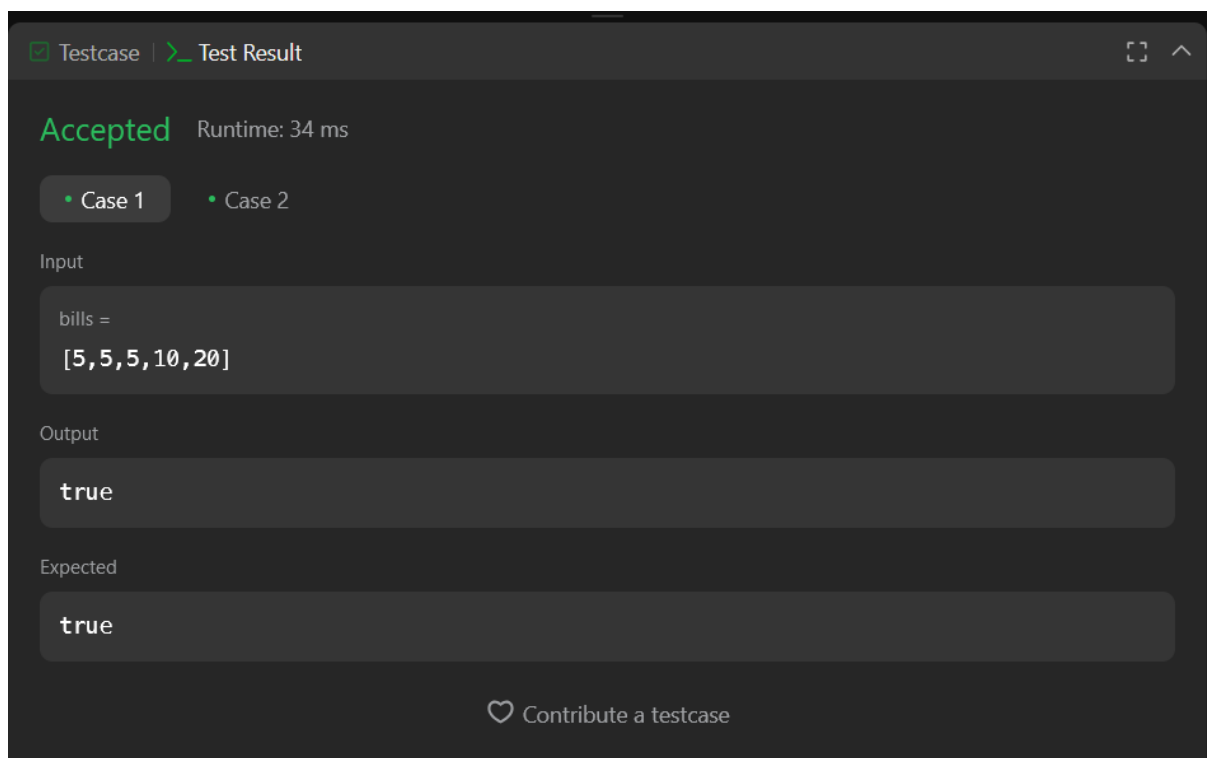
8

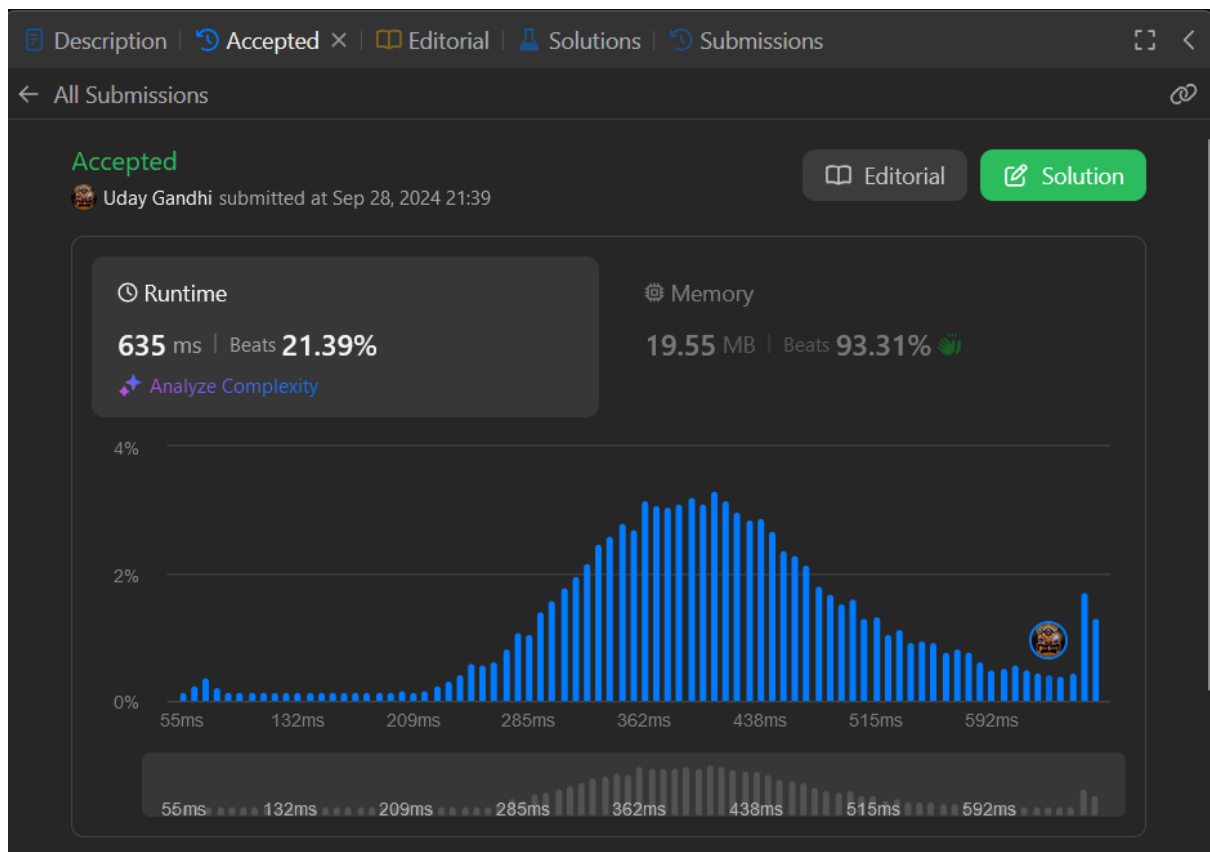
Contribute a testcase



➤ Question 7:**Lemonade Change. Leetcode problem number: 860**➤ Solution:• Source Code:

```
class Solution:
    def lemonadeChange(self, bills: List[int]) -> bool:
        five, ten = 0, 0
        for bill in bills:
            if bill == 5:
                five += 1
            elif bill == 10:
                if five > 0:
                    five -= 1
                    ten += 1
            else:
                return False
            elif bill == 20:
                if ten > 0 and five > 0:
                    ten -= 1
                    five -= 1
                elif five >= 3:
                    five -= 3
            else:
                return False
        return True
```

• Solution:



➤ Question 8:**Merge Intervals Leetcode problem number: 56**➤ Solution:• Source Code:

class Solution:

def merge(self, intervals: List[List[int]]) -&gt; List[List[int]]:

intervals.sort(key=lambda x: x[0])

merged = []

for interval in intervals:

if not merged or merged[-1][1] &lt; interval[0]:

merged.append(interval)

else:

merged[-1][1] = max(merged[-1][1], interval[1])

return merged

• Output:

Testcase | Test Result

**Accepted** Runtime: 37 ms

• Case 1 • Case 2

Input

intervals =

`[[1,3],[2,6],[8,10],[15,18]]`

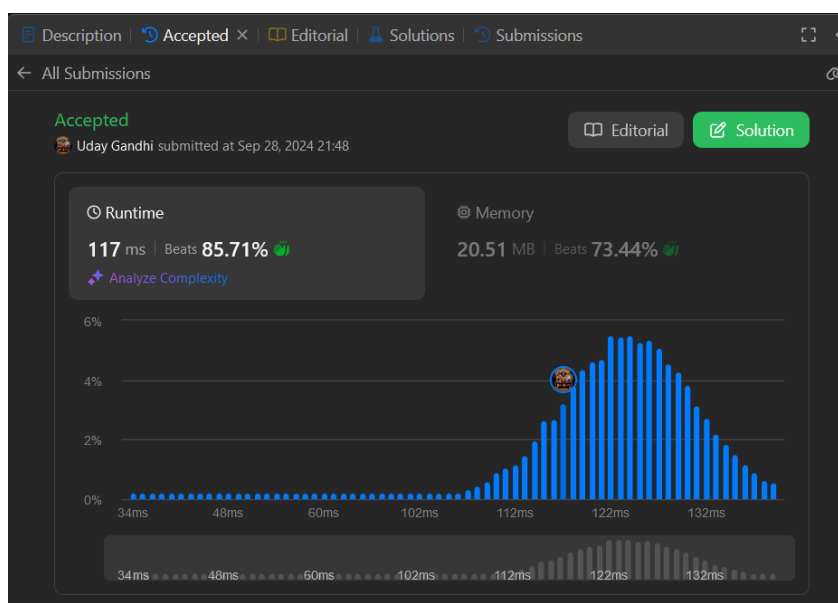
Output

`[[1,6],[8,10],[15,18]]`

Expected

`[[1,6],[8,10],[15,18]]`

Contribute a testcase



➤ Question 9:**LCS LeetCode problem number 1143**➤ Solution:• Source Code:

class Solution:

def longestCommonSubsequence(self, text1: str, text2: str) -&gt; int:

m, n = len(text1), len(text2)

dp = [[0] \* (n + 1) for \_ in range(m + 1)]

for i in range(1, m + 1):

for j in range(1, n + 1):

if text1[i - 1] == text2[j - 1]:

dp[i][j] = dp[i - 1][j - 1] + 1

else:

dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

return dp[m][n]

• Output:

Testcase | Test Result

**Accepted** Runtime: 37 ms

• Case 1 • Case 2 • Case 3

Input

text1 =  
"abcde"

text2 =  
"ace"

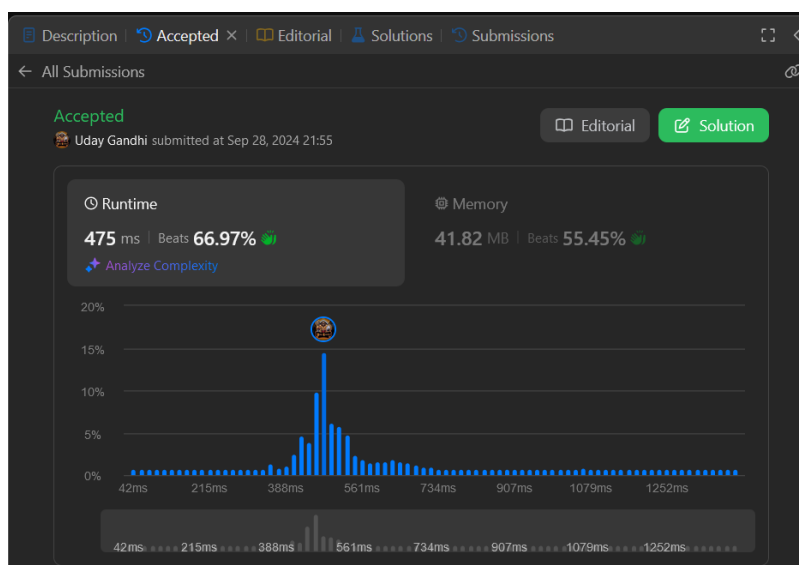
Output

3

Expected

3

♥ Contribute a testcase



➤ Question 10:**Number of Coins GeeksForGeeks**➤ Solution:• Source Code:

```

class Solution:
    def minCoins(self, coins, M, sum):
        k = float("inf")
        dp = [[k for _ in range(sum + 1)] for _ in range(M + 1)]
        dp[0][0] = 0
        for i in range(1, M + 1):
            for j in range(1, sum + 1):
                if coins[i - 1] <= j:
                    dp[i][j] = min(dp[i][j] - coins[i - 1]] + 1, dp[i - 1][j])
                else:
                    dp[i][j] = dp[i - 1][j]
        if dp[M][sum] == k:
            return -1
        return dp[M][sum]

# Driver code
if __name__ == "__main__":
    T = int(input())
    for i in range(T):
        v, m = input().split()
        v, m = int(v), int(m)
        coins = [int(x) for x in input().split()]
        ob = Solution()
        ans = ob.minCoins(coins, m, v)
        print(ans)

```

• Output:

The screenshot shows an IDE's Output Window with a dark theme. At the top, there's a title bar 'Output Window' with standard window controls. Below it, there are two tabs: 'Compilation Results' (which is active and highlighted with a blue underline) and 'Custom Input'. The main content area shows the following text:

```

Compilation Completed

For Input: 30 3
           25 10 5

Your Output:
2

Expected Output:
2

```

The output indicates that the program compiled successfully and produced the correct output for the given input.

Output Window

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully

[Suggest Feedback](#)

Test Cases Passed

**223 / 223**

Attempts : Correct / Total

**1 / 1**

Accuracy : 100%

Points Scored

**4 / 4**

Your Total Score: 8

Time Taken

**0.71**