Competitive Programming

Lab Assignment 01[Week 01]

Name : **R Govardhan Sai Ganesh**

Hall Ticket No : **2303A54044**

Batch : **48**

Day : **Wednesday**

Question 1: Assignment 3: Maximum Profit Streak (Divide and Conquer)

Problem Statement

A company tracks daily profit changes as an array A (values can be negative). A "profit streak" is any non-empty contiguous subarray. Find the maximum possible sum of a profit streak using a divide-and-conquer approach (split into left, right, and crossing subproblems). Input Format The first line contains integer T.For each test case: - First line: N - Second line: N integers A1.

Output Format For each test case, print one integer: maximum subarray sum. Constraints - 1 ≤ T ≤ 20 - 1 ≤ N ≤ 200000 (sum of N over all test cases ≤ 200000) - -10^9 ≤ Ai ≤ 10^9

Sample Input

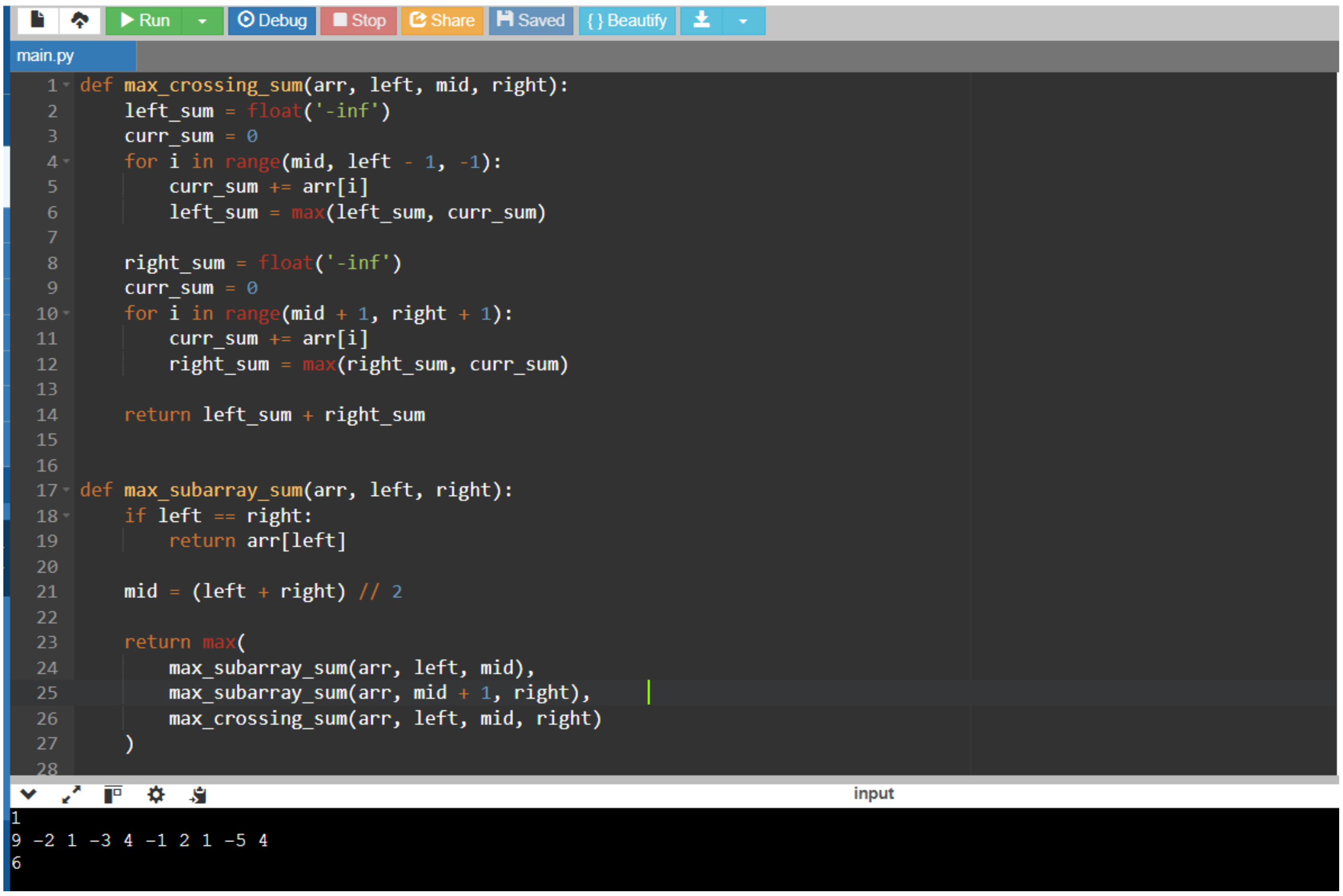
1

9 -2 1 -3 4 -1 2 1 -5 4

Expected Output 6

Code:

```
def max_crossing_sum(arr, left, mid, right):
    left_sum = float('-inf')
    curr_sum = 0
    for i in range(mid, left - 1, -1):
        curr_sum += arr[i]
        left_sum = max(left_sum, curr_sum)
    right_sum = float('-inf')
    curr_sum = 0
    for i in range(mid + 1, right + 1):
        curr_sum += arr[i]
        right_sum = max(right_sum, curr_sum)
    return left_sum + right_sum
def max_subarray_sum(arr, left, right):
    if left == right:
        return arr[left]
    mid = (left + right) // 2
    return max(
        max_subarray_sum(arr, left, mid),
        max_subarray_sum(arr, mid + 1, right),
        max_crossing_sum(arr, left, mid, right)
    )
T = int(input().strip())
for _ in range(T):
    data = list(map(int, input().split()))
    n = data[0]
    arr = data[1:]
    print(max_subarray_sum(arr, 0, n - 1))
```



```
def max_crossing_sum(arr, left, mid, right):
    left_sum = float('-inf')
    curr_sum = 0
    for i in range(mid, left - 1, -1):
        curr_sum += arr[i]
        left_sum = max(left_sum, curr_sum)

    right_sum = float('-inf')
    curr_sum = 0
    for i in range(mid + 1, right + 1):
        curr_sum += arr[i]
        right_sum = max(right_sum, curr_sum)

    return left_sum + right_sum


def max_subarray_sum(arr, left, right):
    if left == right:
        return arr[left]

    mid = (left + right) // 2

    return max(
        max_subarray_sum(arr, left, mid),
        max_subarray_sum(arr, mid + 1, right),
        max_crossing_sum(arr, left, mid, right)
    )

```

```
1
9 -2 1 -3 4 -1 2 1 -5 4
6
```

Question 2:

Problem Statement

You are given N jobs. Each job takes exactly 1 unit of time. Job i has a deadline Di and a profit Pi. If a job is completed on or before its deadline, its profit is earned; otherwise, it cannot be counted. You can perform at most one job at a time. Your task is to choose and schedule jobs to maximize total profit. For each test case, output: (1) the number of jobs completed (2) the maximum total profit.

Input Format The first line contains an integer T, the number of test cases.For each test case: - The first line contains an integer N. - The next N lines each contain two integers Di and Pi.

Output Format For each test case, print two integers:jobs_done total_profit

Constraints - 1 ≤ T ≤ 20 - 1 ≤ N ≤ 200000 (sum of N over all test cases ≤ 200000) - 1 ≤ Di ≤ 100000- 1 ≤ Pi ≤ 10^9

Sample Input

1

5

2 100

1 19

2 27

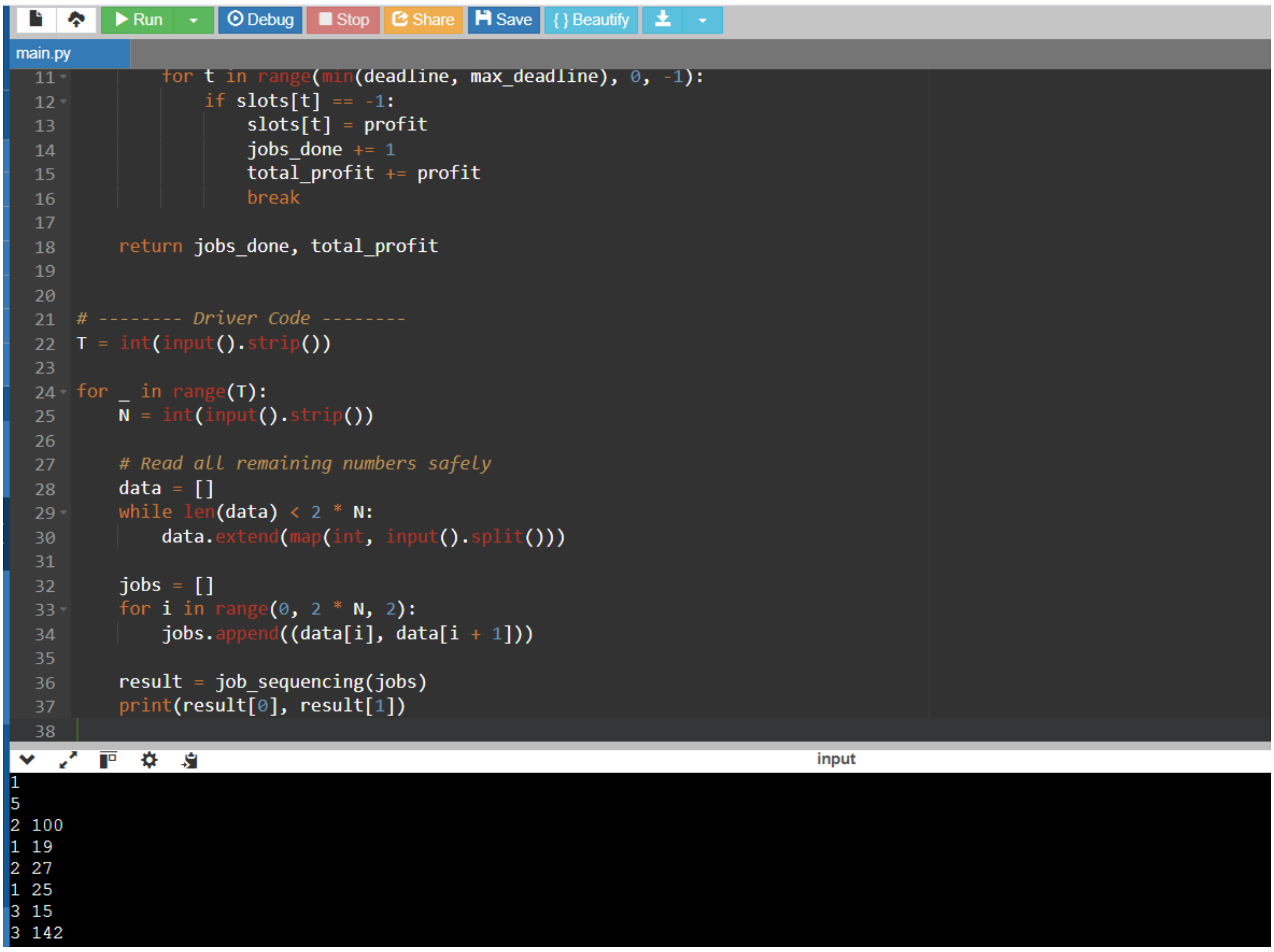
1 25

3 15

Expected Output 3 142

Code:

```
def job_sequencing(jobs):

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

    max_deadline = max(job[0] for job in jobs)

    slots = [-1] * (max_deadline + 1)

    jobs_done = 0

    total_profit = 0

    for deadline, profit in jobs:

        for t in range(min(deadline, max_deadline), 0, -1):

            if slots[t] == -1:

                slots[t] = profit

                jobs_done += 1
            total_profit += profit
            break
    return jobs_done, total_profit
T = int(input().strip())
for _ in range(T):
    N = int(input().strip())
    data = []
    while len(data) < 2 * N:
        data.extend(map(int, input().split()))
    jobs = []
    for i in range(0, 2 * N, 2):
        jobs.append((data[i], data[i + 1]))
    result = job_sequencing(jobs)
    print(result[0], result[1])
```



```
        for t in range(min(deadline, max_deadline), 0, -1):
            if slots[t] == -1:
                slots[t] = profit
                jobs_done += 1
                total_profit += profit
                break

    return jobs_done, total_profit


# -------- Driver Code --------
T = int(input().strip())

for _ in range(T):
    N = int(input().strip())

    # Read all remaining numbers safely
    data = []
    while len(data) < 2 * N:
        data.extend(map(int, input().split()))

    jobs = []
    for i in range(0, 2 * N, 2):
        jobs.append((data[i], data[i + 1]))

    result = job_sequencing(jobs)
    print(result[0], result[1])

```

```
1
5
2 100
1 19
2 27
1 25
3 15
3 142
```