

# Competitive Programming

## Lab Assignment 01[Week 01]

---

Name :BG.Sreevani  
Hall Ticket No :2303A54066  
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 -  $1 \leq T \leq 20$  -  $1 \leq N \leq 200000$  (sum of N over all test cases  $\leq 200000$ ) -  $-10^9 \leq A_i \leq 10^9$

##### 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))

```

The screenshot shows the OnlineGDB web interface. The left sidebar displays user information and navigation links. The main workspace shows Python code for finding the maximum subarray sum. The code uses dynamic programming to handle three cases: crossing the middle element, and two recursive calls for left and right halves. The code is run with test input [1, -2, 1, -3, 4, -1, 2, 1, -5, 4] and produces the output '6'.

```

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)
    )

```

input

```

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

```

...Program finished with exit code 0

## Question 2:

### Assignment 3: Job Sequencing with Deadlines (Greedy)

#### 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** -  $1 \leq T \leq 20$  -  $1 \leq N \leq 200000$  (sum of N over all test cases  $\leq 200000$ ) -  $1 \leq D_i \leq 100000$  -  $1 \leq P_i \leq 10^9$

#### **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])

```

The screenshot shows the onlinegdb.com web interface. The left sidebar includes navigation links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. A banner at the bottom left from auth0 says 'You asked, we delivered! Our Free Plan now includes a Custom Domain, 5 Actions, and 25,000 MAUs.' The main area has tabs for 'Run' (selected), 'Debug', 'Stop', 'Share', 'Saved', and 'Beautify'. The code editor contains the provided Python script. Below the editor is a terminal window labeled 'input' containing the sequence of numbers: 1, 5, 2, 100, 1, 19, 2, 27, 1, 25, 3, 15, 3, 142.

```

main.py
1 def job_sequencing(jobs):
2     jobs.sort(key=lambda x: x[1], reverse=True)
3     max_deadline = max(job[0] for job in jobs)
4     slots = [-1] * (max_deadline + 1)
5     jobs_done = 0
6     total_profit = 0
7     for deadline, profit in jobs:
8         for t in range(min(deadline, max_deadline), 0, -1):
9             if slots[t] == -1:
10                 slots[t] = profit
11                 jobs_done += 1
12                 total_profit += profit
13                 break
14     return jobs_done, total_profit
15 T = int(input().strip())
16 for _ in range(T):
17     N = int(input().strip())
18     data = []
19     while len(data) < 2 * N:
20         data.extend(map(int, input().split()))
21     jobs = []
22     for i in range(0, 2 * N, 2):
23         jobs.append((data[i], data[i + 1]))
24     result = job_sequencing(jobs)
25     print(result[0], result[1])

```

input

```

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

```